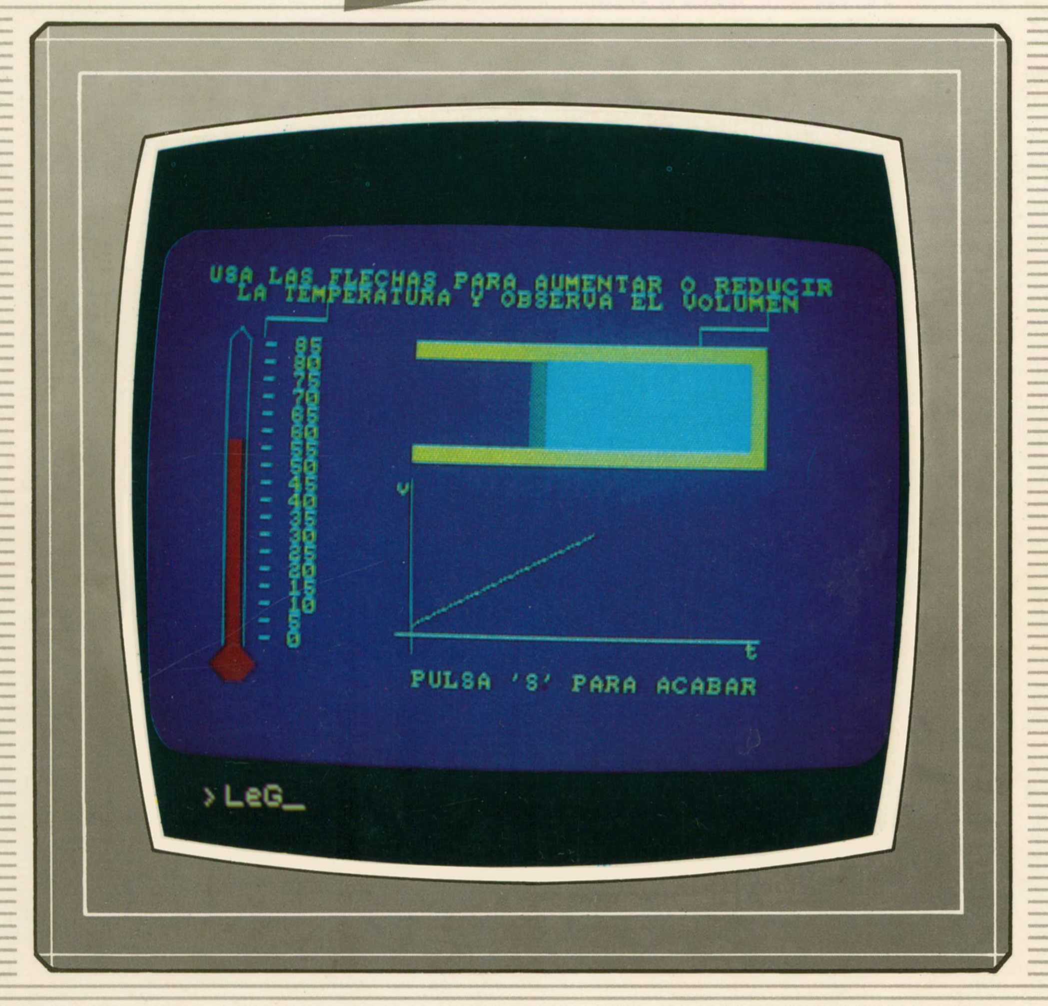
▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 18

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 18

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.

Diseño y maquetación:
BRAVO-LOFISH.

Fotografía:
EQUIPO GALATA.

Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Sociedad Tamariz, Diplomada en Telecomunicación. OTROS LENGUAJES (COBOL): Eloy Pérez, Licenciado en Informática. Ana Pastor, Licenciada en Informática.

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-140-1
ISBN de la obra: 84-7688-068-7

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-5-677-1987
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Julio, 1987.
P.V.P. Canarias: 335,–.

INDICE

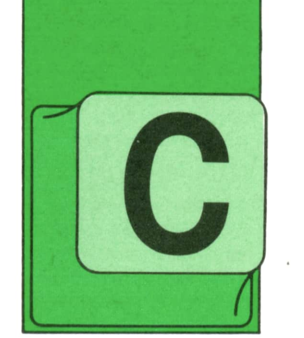


INFORMATICA BASICA

LENGUAJES DE PROGRAMACION



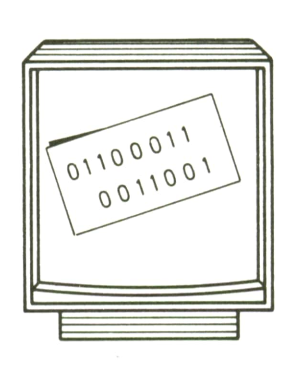
Introducción

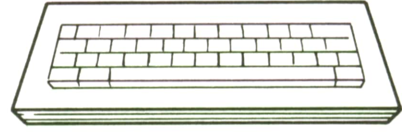


CON la aparición en el mercado de los computadores de programas almacenados, se creó una nueva profesión, el programador de ordenadores. Desde los comienzos hasta hoy, se han producido unos enormes avances en los diferentes tipos y formas de programar.

Son muchos los que atribuyen el honor de haber sido la primera programadora a Ada Augusta, que murió casi un siglo antes de que apareciera la primera computadora programable. Nació en 1815. Ada se relacionó con Charles Babbage, quien trabajaba en su proyecto de la máquina analítica. Ada, que poseía grandes aptitudes para las matemáticas y el pensamiento mecánico, se ofreció para trabajar con Babbage, en su proyecto, y en 1842 tradujo del inglés al italiano una primera descripción de la citada máquina, agregando muchas notas propias. Ada se refirió a «ciclos de operación» y al repetido uso de las tarjetas como estructuras del tipo subrutinas y se refirió también a la computación del tipo no numérica. Observó que la máquina analítica no originaba nada, y que sólo podría hacer aquello que se le ordenaba.

Estos fueron los primeros esbozos de la programación. Veamos ahora los principios reales.



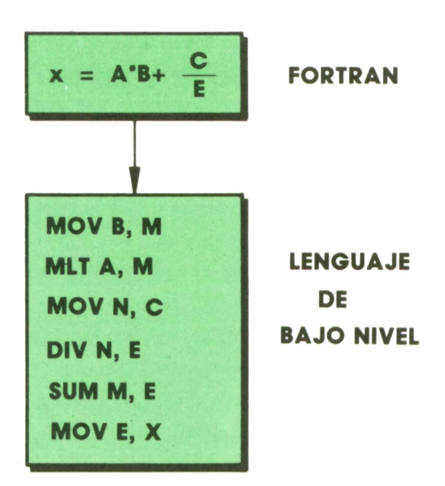




El lenguaje máquina, por estar íntimamente ligado al hardware del ordenador, es propio de cada uno.

Los inicios

Los lenguajes de programación surgieron como canal de comunicación entre el hombre y la máquina; estos lenguajes permiten que la máquina ejecute las órdenes dadas por el programador. Los lenguajes de programación toman diversas formas. Los primeros pertenecientes a las primeras máquinas, como la ENIAC y la EDSAC, se componían en el lenguaje real de las máquinas mismas. Las instrucciones se expresan simplemente como una serie de dígitos binarios. La gran dificultad que suponía este tipo de programación condicionaba mucho la capacidad de estas primeras máquinas.



En los lenguajes de alto nivel se permite ejecutar en una sola instrucción operaciones que equivalen a varias instrucciones en lenguaje tipo ensamblador.

Los primeros lenguajes de programación propiamente dichos se conocieron con el nombre de lenguajes ensambladores; un ejemplo es el TRANSCODE. En los lenguajes ensambladores se define un código especial (llamado mnemónico) para cada una de las operaciones de la máquina y se introduce con una notación especial, para especificar el dato con el cual debe realizarse tal operación.

Estos lenguajes son aún muy populares en ciertas aplicaciones; a pesar de que se ha avanzado notablemente en los lenguajes de programación, de máquina, esto no basta para satisfacer las necesidades de todo lo que el programador desea hacer.



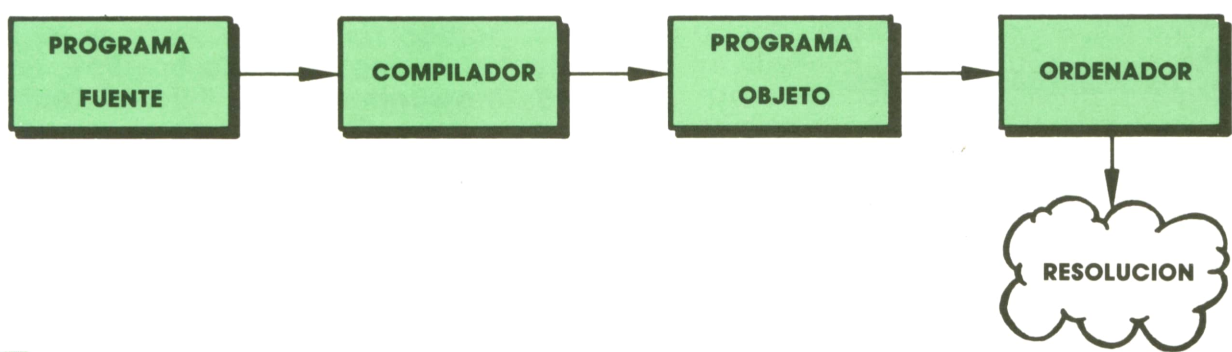
Lenguajes de alto nivel

A mediados del decenio 1950-1960 aparecieron los lenguajes de programación de propósito general, uno de los cuales revolucionó rápidamente el campo de la programación. Se llamó FORTRAN (FORmula TRANSlation) y fue publicado en 1954.

El líder del proyecto FORTRAN fue John Backus, quien trabajó para la IBM y desarrolló un método formal, para definir la sintaxis de los lenguajes de programación, la forma BNF. FORTRAN sufrió varios desarrollos posteriores que fueron apareciendo en 1958, 1960 y 1962, el último de los cuales se conoció como FORTRAN IV. En 1977 apareció el FORTRAN 77.

Se trata de un lenguaje dirigido a la solución numérica de problemas científicos, es fácil de entender, leer y escribir. Con el FORTRAN, el usuario está capacitado de inmediato para escribir un programa aunque sepa muy poco acerca de las características físicas de la máquina.

Los lenguajes de alto nivel, a causa de su generalidad, requieren traductores más complejos, conocidos como «compiladores». Lo cual produjo a estos lenguajes algunas dificultades al principio.





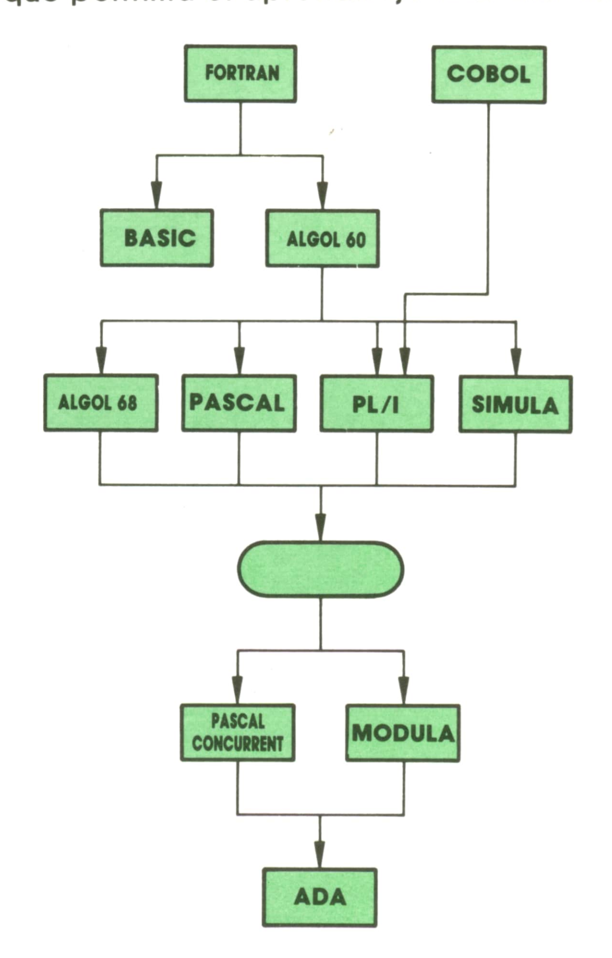
Relación entre el programa FUENTE y el programa OBJETO.

Otros lenguajes de programación han seguido los pasos del FORTRAN; tal es el caso del ALGOL (ALGOrithmic Languaje). Fue diseñado por un comité internacional en 1958 y revisado en 1960. Es un lenguaje muy efectivo para resolver problemas matemáticos y numéricos.

Tanto FORTRAN como ALGOL están dirigidos básicamente a la computación científica. En mayo de 1959 el Departamento de Defensa de los Estados Unidos convocó una reunión para discutir el problema de desarrollar un lenguaje común, para aplicaciones de negocios. La ver-

sión oficial de COBOL (COmmon Business Oriented Languaje, Lenguaje Orientado a los Negocios) apareció en diciembre de 1959.

Los objetivos de COBOL consideraban la expresión natural de los programas, lo que permitía el aprendizaje fácil del lenguaje, la amplia documentación del mismo y la independencia de la máquina, lo cual facilitaría la transferencia de los programas COBOL de una máquina a otra.



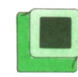

Lenguajes de programación.

Han surgido muchas variaciones de COBOL, sin embargo, básicamente no ha sufrido cambio alguno.

Otro lenguaje de alto nivel que tuvo gran repercusión fue el BASIC (Beginner's All-purpose Symbolic Instruction Code). Es un lenguaje de alto nivel orientado al aprendizaje y de contexto científico. El sistema BASIC fue desarrollado en 1965, y fue el primero en estar disponible en tiempo compartido o modo interactivo y en utilizarse en una red o base distribuida. Continúa siendo hoy en día de gran difusión, y el tiempo compartido ha sido la forma más común de utilización.

En septiembre de 1963 un comité compuesto por personal de IBM y de clientes formó un consejo con el fin de crear un lenguaje que pudiera atraer a más usuarios, pero que siguiese siendo también una poderosa herramienta para el ingeniero. El informe presentado fue revisado en junio y diciembre del mismo año y denominado PL/I. Por ser un lenguaje muy general tiene una amplísima variedad de aplicaciones, pero aunque es un lenguaje muy potente, no pudo desempeñar el papel que se esperaba debido a que no cumplía el requisito de universalidad.

Otro lenguaje que ha tenido hoy en día una gran difusión en todas sus variantes ha sido el PASCAL, que proviene del ALGOL en su variante ALGOL 60. El PASCAL es un poderoso lenguaje científico para tratamiento de todo tipo de datos matemáticos. Su principal ventaja radica en que es un lenguaje estructurado, facilitando la programación modular.



Lenguajes de inteligencia artificial

Los lenguajes de inteligencia artificial están teniendo un gran desarrollo en la actualidad. El más conocido y del que han partido todos los demás es el LISP (LISt Processing). Creado en 1959, en el MIT (Massachusetts Institute of Technology) para aplicaciones en el campo de la inteligencia artificial, es el preferido por los programadores en este campo. En este lenguaje las diferencias entre los datos y las instrucciones son casi imperceptibles y tiene gran facilidad para la programación.

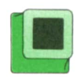
Otros lenguajes en este campo son:

LOGO. Procedente del LISP, pensado principalmente para la enseñanza escolar. Fue desarrollado en el MIT.

APL. Creado por IBM; permite con una única instrucción definir operaciones para las cuales otros ordenadores emplean varias.

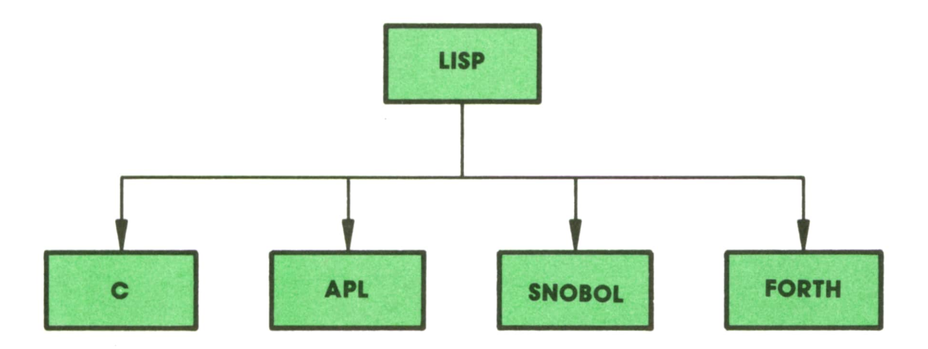
FORTH. Desarrollado para control de proceso industrial

C. Creado para poder utilizar al máximo de las facilidades del sistema operativo UNIX.



Lenguajes actuales de mayor profusión

En la actualidad, el lenguaje más poderoso que se ha creado, y que pretende esa universalidad buscada por PL/I anteriormente, es el ADA. El ADA es quizá el lenguaje de programación más reciente. Fue creado en 1975 a partir de una modalidad del PASCAL, llamada PASCAL CONCURRENTE y de otro lenguaje llamado MODULA. Las propiedades esenciales de ambos están recogidas en el ADA; está enfocado de manera que las diferentes partes de un programa se consideran como componentes, que se pueden seleccionar por medio de una especie de catálogos y más tarde combinar en función de la aplicación que se quiera. A estos componentes se les llama módulos y el compilador ADA los conecta entre sí.

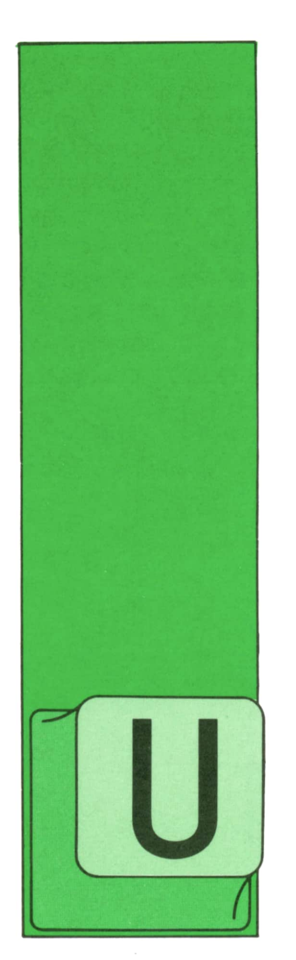




Lenguajes de inteligencia artificial.

MAQUINA Z-80

SPECTRUM, AMSTRAD, MSX





Las subrutinas y el stack

UNA de las estructuras de datos más usada es la pila, o stack. Concretamente el Z-80, como casi todos los microprocesadores, tiene una pila del sistema para gestionar las llamadas a subrutinas.

Estructura de una pila

Una pila, como su nombre indica, es una zona donde se «apilan» datos. Cada nuevo dato se coloca «encima» del anterior; por eso esta estructura también es conocida como LIFO (Last In First Out = último en llegar, primero en salir).

Para implementarla en la memoria de un ordenador se designa una cierta posición de memoria como «fondo de la pila» y en los siguientes se van almacenando consecutivamente los datos. Para tener la cuenta de cuál es el último dato

MANEJO DE LA PILA DEL SISTEMA

Código mnemotécnico	Operación simbólica	Indicadores S Z H P/V N C	Códigos 76	543	210	Hex	N.º de bytes	N.º de ciclos M	N.º de estados T	Comentarios
PUSH qq	$(SP-2) \leftarrow qq_L$	• • X • X • • •	11	qq0		1	3	11	qq	Par
	$(SP-1) \leftarrow qq_H$								00	BC
	$SP \rightarrow SP-2$								01	DE
PUSH IX	$(SP-2) \leftarrow IX_L$	• • X • X • • •	11	011	DD	2	4	15	10	HL
	$(SP-1) \rightarrow IX_H$		11	100	E5				11	AF
	$SP \rightarrow SP-2$									
PUSH IY	$(SP-2) \leftarrow IY_L$	• • X • X • • •	11	111	FD	2	4	15		
	$(SP-1) \leftarrow IY_H$		11	100	E5					
	$SP \rightarrow SP-2$									
POP qq	$qq_H \leftarrow (SP+1)$	• • X • X • • •	11	qq0		1	3	10		
	$qq_L \leftarrow (SP)$									
	$SP \rightarrow SP+2$									
POP IX	$IX_H \leftarrow (SP+1)$	• • X • X • • •	11	011	DD	2	4	14		
	$IX_L \leftarrow (SP)$		11	100	E1					
	$SP \rightarrow SP+2$									
POP IY	$IY_H \leftarrow (SP+1)$	• • X • X • • •	11	111	FD	2	4	14		
	$IY_L \leftarrow (SP)$		11	100	E1					
	$SP \rightarrow SP+2$									
EX (SP),HL	$H \leftrightarrow (SP+1)$	• • X • X • • •	11	100	E3	1	5	19		
	$L \leftrightarrow (SP)$									
EX (SP),IX	$IX_H \leftrightarrow (SP+1)$	• • X • X • • •	11	011	DD	2	6	23		
	$IX_L \leftrightarrow (SP)$		11	100	E3					
EX (SP),IY	$IY_H \leftrightarrow (SP+1)$	• • X • X • • •	11	111	FD	2	6	23		
	$IY_L \leftrightarrow (SP)$		11	100	E3					

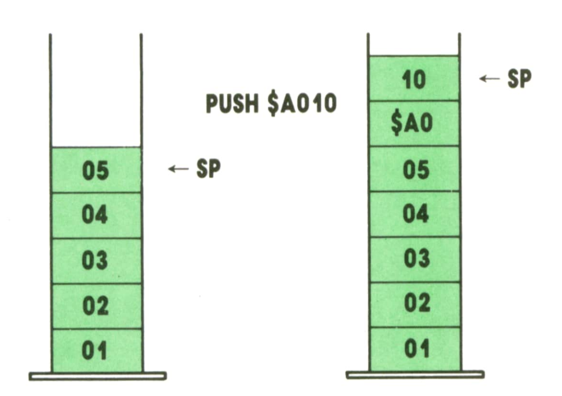
se utiliza un registro que nos lo indica; éste es el puntero de la pila SP (Stack Pointer).

Por tanto, para introducir un dato en la pila el Z-80 incrementa el puntero SP y en la posición indicada por éste almacena el dato.

Para realizar la operación contraria coge el dato indicado por el SP y decrementa el contenido de éste en una unidad.

Como las direcciones del Z-80 son de 16 bits y la pila sólo admite datos de 8 bits, éstos se almacenan en forma de dos datos de 8 bits consecutivos, por lo que se incrementará, o decrementará, el puntero de la pila en dos unidades.

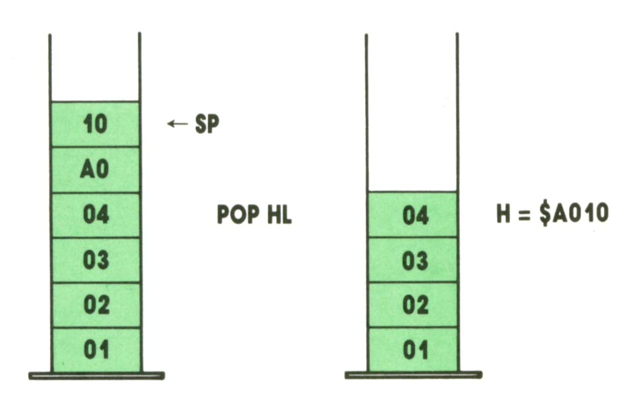
Para guardar los datos en la pila se usa la instrucción PUSH (empujar); pudiendo ser el dato cualquiera de los dobles registros de 16 bits (AF, BC, DE, HL, IX, IY).





Ejecución de PUSH.

La instrucción inversa de ésta es la que saca los datos de la pila y los almacena en cualquiera de los registros. La instrucción es POP, y también funciona con los mismos registros que PUSH.





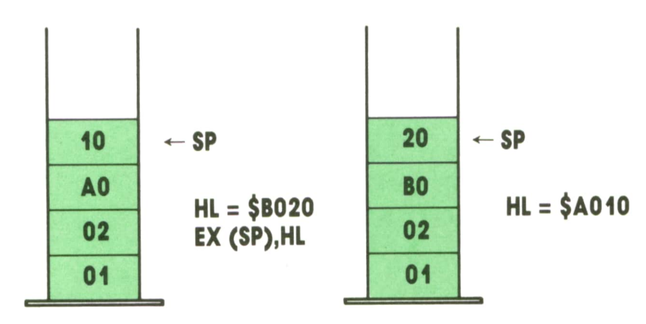
Ejecución de POP.

También existen para manejo de pila algunas instrucciones de intercambio de algunos registros con los contenidos de dicha pila. Como se vio en un anterior capítulo, la instrucción EX intercambia los contenidos de las direcciones, o registros, sobre los que opere.

En este caso las instrucciones existentes son:

EX (SP),HL
EX (SP),IX
EX (SP),IY

que intercambia el contenido de la pila con el contenido de los registros, HL,IX e IY, respectivamente.





Ejecución de EX.



Llamada y retorno de subrutinas

Cuando en un programa existe una parte que se repite en varias ocasiones, ésta se suele extraer como una subrutina que se llama cada vez que se necesita, en vez de repetir el código completo, por lo que se ahorra espacio.

Una vez ejecutada la subrutina la ejecución del programa debe continuar en la instrucción siguiente a la que llamó la subrutina. Por ello, es necesario almacenar la dirección desde donde se ejecuta la llamada.

Como una subrutina puede llamar a otra subrutina, es necesario almacenar más de una dirección de retorno. Para esto se usa la pila del sistema anteriormente descrita. Este sistema permite ejecutar los retornos en el orden inverso al que se ejecutaron las llamadas.

La instrucción para ejecutar una subrutina es:

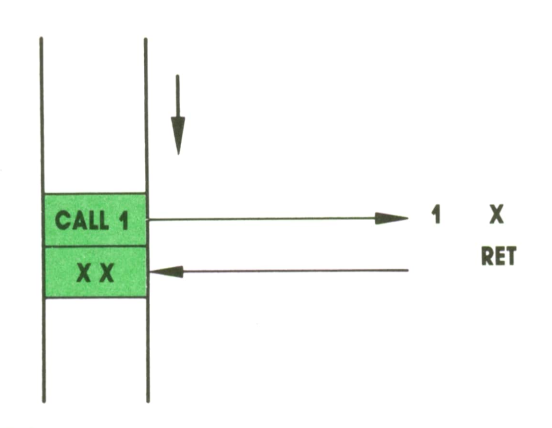
CALL X

GRUPO DE LLAMADA Y RETORNO

Código mnemotécnico	Operación simbólica	Indicadores S Z H P/V N C	Códigos 76 543 210	Hex	N.º de bytes	N.º de ciclos M	N.º de estados T	Comentarios
CALL nn	(SP − 1) ← PC_H (SP − 2) ← PC_L PC ← nn	• • X • X • • •	11 001 101 ← n → ← n →	CD	3	5	17	
CALL cc,nn	Si la condición cc es falsa, se continúa; si es cierta, como CALL nn	• • X • X • • •	11 cc 100 ← n → ← n →		3 3	3 5	10 17	Si cc es falsa Si cc es cierta
RET	PC_L ← (SP) PC_H ← (SP + 1)	• • X • X • • •	11 001 001	C9	1	3	10	
RET cc	Si la condición cc es falsa, se continúa; si no, como RET	• • X • X • • •	11 cc 000		1 1	1 3	5 11	Si cc es falsa Si cc es cierta cc Condición 000 NZ no cero 001 Z cero 010 NC no arrastre 011 C arrastre 100 PO paridad impar 101 PE paridad par 110 P signo positivo 111 M signo negativo

donde X es la dirección absoluta donde está la subrutina.

Al ejecutar esta instrucción el contador de programa (PC) se guarda en la pila para después regresar a seguir la ejecución, tras esto se ejecutan las instrucciones que están a partir de la dirección indicada.





Llamada a subrutina.

Para regresar de esta subrutina debemos ejecutar la instrucción:

RET

ésta saca la dirección que estaba ejecutándose al realizarse la llamada a la subrutina de la pila del sistema y continúa la ejecución a partir de este punto como si la llamada no se hubiese realizado.

Existen algunas instrucciones de salto a subrutina condicionales que se ejecutan, o no se ejecutan, según el estado de algunas de las banderas.

Si la condición indicada en la instrucción es falsa, la instrucción no se ejecutará. Si la condición es cierta, funciona igual que CALL.

Las diversas condiciones son:

NZ: El contenido del acumulador es distinto de cero.

Z: El contenido del acumulador es cero.

NC: En la última operación no se ha producido acarreo.

C: En la última operación se ha producido acarreo.

PO: Paridad impar (el número de «1» en el acumulador es impar).

PE: Paridad par (el número de «1» en el acumulador es par).

P: El número almacenado en el acumulador es positivo.

M: El número almacenado en el acumulador es negativo.

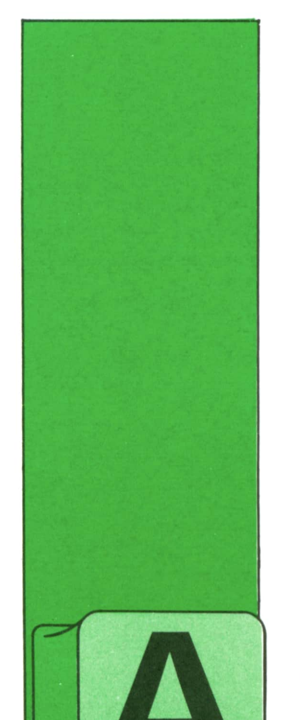
También existe ejecución condicional de la instrucción de retorno. Esta lleva las mismas opciones que la de llamada.

PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS



Diseñador de juegos de aventuras



continuación aparece un programa realmente excepcional. Con él podremos realizar nuestros propios programas de aventuras.

Las aventuras que podremos realizar, aunque pueden ser englobadas dentro de las llamadas conversacionales, no lo son del todo. Aun así, los juegos que nosotros podremos realizar podrán ser todo lo espectaculares y complicados que queramos.

Antes de ver el listado y las modificaciones que hay que realizar para que el programa funcione en ordenadores distintos del IBM, vamos a explicar un poco en qué consiste una de estas aventuras y cómo se crearía.

Todas las aventuras conversacionales están basadas en un laberinto, en el cual nos van a ocurrir ciertas cosas. Dentro de dicho laberinto o paisaje tendremos que llevar a cabo una misión que puede consistir en rescatar a una persona, capturar o matar a un malvado rey o monstruo, intentar escapar, desactivar una bomba, etcétera. El laberinto estára dividido en habitaciones o, en forma general, en lugares. En cada uno de estos lugares podrá pasar algo, como puede ser elegir un camino, coger un objeto, usar un objeto, pelearnos con alguien, etc. Dichos lugares están interconectados unos con otros de una forma lógica y que nosotros hemos inventado. La aventura empieza en una de las habitaciones y puede terminar en más de una, dependiendo de que el jugador termine la aventura vivo o muerto (o prisionero de por vida).

Con este programa podremos definir todos los lugares que queramos, qué podemos hacen en cada lugar, qué necesitamos utilizar en ciertos lugares, dónde terminamos y cómo terminamos la aventura...

Antes de pasar a una aventura de ejemplo vamos a ver el programa:

```
1000 REM ***************************
1010 REM * DISENADOR DE AVENTURAS *
1020 REM ***************************
1030 REM
1040 REM **********************
1050 REM *       AUTORES      *
1060 REM *       -------      *
1070 REM * MANUEL  ALFONSECA *
1080 REM *          Y         *
1090 REM * FRANCISCO MORALES *
1100 REM **********************
1110 REM
1120 REM ********************************
1130 REM **** (c) Ed. Siglo Cultural ****
1140 REM **** (c) 1987               ****
1150 REM ********************************
1160 REM
```

```
1170 REM *****************************
1180 REM * DEFINICION DE LAS TABLAS *
1190 REM *****************************
1200 REM
1210 DIM S$(300)
1220 DIM E$(800)
1230 DIM O$(100)
1240 DIM CL(300)
1250 DIM IN(300)
1260 DIM E1(300)
1270 DIM O1(300)
1280 DIM V1(800)
1290 DIM V2(250)
1300 DIM V3(100)
1310 DIM V4(170)
1320 DIM HB(100)
1330 DIM FZ(100)
1340 DIM PR(100)
1350 DIM CN(100)
1360 REM
1370 REM ***************************
1380 REM * DEFINICION DE VARIABLES *
1390 REM ***************************
1400 REM
1410 LET I=0 : REM INDICE DE SITIOS
1420 LET IE=0: REM INDICE DE CAMINOS
1430 LET IL=0: REM INDICE DE LISTA DE CAMINOS
1440 LET IM=0: REM INDICE DE ENEMIGOS
1450 LET IO=0: REM INDICE DE OBJETOS
1460 LET IJ=0: REM INDICE DE LISTA DE OBJETOS
1470 LET IC=0: REM INDICE DE CONDICIONES
1480 REM
1490 REM *** BUCLE DE CREACION ***
1500 REM
1510 CLS
1520 PRINT " LUGAR No.";I+1
1530 PRINT
1540 PRINT " Pulsa CONTROL-Q para que en este"
1550 PRINT "lugar se borre la pantalla y CON-"
1560 PRINT "TROL-P para saltar de linea."
1570 PRINT " Pulsa ENTER para terminar."
1580 PRINT
1590 PRINT " (Que te dicen?"
1600 PRINT
1610 LINE INPUT A$
1620 IF A$="" THEN GOTO 2960
1630 LET S$(I+1)=A$
1640 LET Y=CSRLIN+1
1650 PRINT
1660 PRINT "(De que clase es?"
1670 PRINT
1680 PRINT "1: Elegir camino"
1690 PRINT "2: Luchar"
1700 PRINT "3: Obtener objetos"
1710 PRINT "4: Comprobar objetos"
1720 PRINT "5: Fin del juego"
1730 PRINT
1740 PRINT "OPCION = ";CHR$(219);CHR$(29);
1750 LET A$=INPUT$(1)
1760 IF A$<"1" OR A$>"5" THEN GOTO 1750
1770 PRINT A$;
1780 FOR Z=1 TO 400
1790 NEXT Z
1800 LET CL(I+1)=VAL(A$)
1810 ON CL(I+1) GOTO 1830,2100,2300,2640,2900
1820 REM
1830 REM *** ELEGIR CAMINO ***
1840 REM
1850 LET IN(I+1)=IL+1
```

```
1860 LET K=0
1870 LOCATE Y,1
1880 FOR Z=1 TO 23-Y
1890    PRINT SPACE$(40)
1900 NEXT Z
1910 LOCATE Y,1
1920 PRINT " Direccion No.";K+1
1930 PRINT
1940 PRINT " Pulsa ENTER para terminar."
1950 PRINT
1960 INPUT B$
1970 IF B$="" THEN GOTO 2060
1980 LET E$(IE+1)=B$
1990 PRINT
2000 PRINT "(A que numero de habitacion"
2010 INPUT "nos lleva";A
2020 LET V1(IE+1)=A
2030 LET IE=IE+1
2040 LET K=K+1
2050 GOTO 1870
2060 LET E1(IL+1)=K
2070 LET IL=IL+1
2080 GOTO 2930
2090 REM
2100 REM *** LUCHAR ***
2110 REM
2120 LET IN(I+1)=IM+1
2130 LOCATE Y,1
2140 FOR Z=1 TO 23-Y
2150    PRINT SPACE$(40)
2160 NEXT Z
2170 LOCATE Y,1
2180 INPUT "(Habilidad del enemigo";HB(IM+1)
2190 PRINT
2200 INPUT "(Fuerza del enemigo";FZ(IM+1)
2210 PRINT
2220 INPUT "(A Que habitacion vas si ganas";V2(3*IM+1)
2230 PRINT
2240 INPUT "(Y si abandonas";V2(3*IM+2)
2250 PRINT
2260 INPUT "(Y si pierdes";V2(3*IM+3)
2270 LET IM=IM+1
2280 GOTO 2930
2290 REM
2300 REM *** OBTENER OBJETOS ***
2310 REM
2320 PRINT
2330 LET IN(I+1)=IJ+1
2340 LET K=0
2350 LOCATE Y,1
2360 FOR Z=1 TO 24-Y
2370    PRINT SPACE$(40)
2380 NEXT Z
2390 LOCATE Y,1
2400 PRINT "Objeto";K+1
2410 PRINT
2420 PRINT " Pulsa ENTER para terminar."
2430 PRINT
2440 INPUT B$
2450 PRINT
2460 IF B$="" THEN GOTO 2570
2470 LET O$(IO+1)=B$
2480 PRINT "Si te lo regalan pulsa 0"
2490 PRINT "Si lo has de comprar pulsa su precio"
2500 PRINT "Si lo tienes que coger tu, pulsa -1"
2510 PRINT
2520 INPUT "(Cuanto cuesta";PR(IO+1)
2530 PRINT
2540 LET IO=IO+1
```

```
2550 LET K=K+1
2560 GOTO 2350
2570 LET O1(IJ+1)=K
2580 PRINT
2590 INPUT "(A donde vas despues";V3(IJ+1)
2600 PRINT
2610 LET IJ=IJ+1
2620 GOTO 2930
2630 REM
2640 REM *** COMPROBAR OBJETOS ***
2650 REM
2660 LET IN(I+1)=IC+1
2670 LOCATE Y,1
2680 FOR Z=1 TO 23-Y
2690    PRINT SPACE$(40)
2700 NEXT Z
2710 LOCATE Y,1
2720 PRINT "(Que objeto tienes que tener";
2730 INPUT B$
2740 FOR K=1 TO IO
2750    IF O$(K)=B$ THEN GOTO 2810
2760 NEXT K
2770 PRINT
2780 PRINT "Cuidado. Todavia no lo has definido"
2790 PRINT "No te olvides hacerlo."
2800 PRINT
2810 LET CN(IC+1)=K
2820 PRINT
2830 INPUT "(A donde vas si lo tienes";V4(2*IC+1)
2840 PRINT
2850 INPUT "(A donde vas si no lo tienes";V4(2*IC+2)
2860 LET IC=IC+1
2870 LET I=I+1
2880 GOTO 1490
2890 REM
2900 REM *** SITIO FINAL DE JUEGO ***
2910 REM
2920 LET IN(I+1)=0
2930 LET I=I+1
2940 GOTO 1490
2950 REM
2960 REM *** GUARDAR EL JUEGO EN CINTA O DISCO ***
2970 REM
2980 CLS
2990 PRINT " Prepara la cinta o el disco para"
3000 PRINT "grabar la aventura."
3010 PRINT
3020 PRINT "PULSA UNA TECLA"
3030 LET A$=INPUT$(1)
3040 PRINT
3050 INPUT "(Nombre de la aventura";F$
3060 IF F$="" THEN GOTO 2980
3070 PRINT
3080 PRINT "GRABANDO ... ";F$
3090 PRINT
3100 PRINT "Espera un momento"
3110 OPEN F$ FOR OUTPUT AS #1
3120 PRINT #1,I;IE;IL;IM;IO;IJ;IC
3130 FOR K=1 TO I
3140    PRINT #1,S$(K)
3150 NEXT K
3160 FOR K=1 TO I
3170    PRINT #1,CL(K);IN(K)
3180 NEXT K
3190 FOR K=1 TO IE
3200    PRINT #1,E$(K)
3210 NEXT K
3220 FOR K=1 TO IE
3230    PRINT #1,V1(K)
3240 NEXT K
```

```
3250 FOR K=1 TO IL
3260     PRINT #1,E1(K)
3270 NEXT K
3280 FOR K=1 TO IM
3290     PRINT #1,HB(K);FZ(K);V2(3*K-2);V2(3*K-1);V2(3*K)
3300 NEXT K
3310 FOR K=1 TO IO
3320     PRINT #1,O$(K)
3330 NEXT K
3340 FOR K=1 TO IO
3350     PRINT #1,PR(K)
3360 NEXT K
3370 FOR K=1 TO IJ
3380     PRINT #1,O1(K);V3(K)
3390 NEXT K
3400 FOR K=1 TO IC
3410     PRINT #1,CN(K);V4(2*K-1);V4(2*K)
3420 NEXT K
3430 CLOSE #1
3440 PRINT
3450 PRINT " Aventura grabada."
3460 PRINT
3470 PRINT " Utiliza el programa GAME2 para"
3480 PRINT "ejecutarla y el programa GAME3"
3490 PRINT "para imprimirla."
3500 PRINT
3510 PRINT " Pulsa 'T' para terminar"
3520 LET A$=INPUT$(1)
3530 IF A$<>"T" AND A$<>"t" THEN GOTO 3520
3540 CLS
3550 PRINT "Adios ..."
3560 PRINT:PRINT:PRINT
```

El programa no ha sido preparado para SPECTRUM, pero funciona perfectamente en el IBM; para AMSTRAD y MSX hay que realizar los siguientes cambios:

AMSTRAD:

```
1640 LET Y=VPOS
1750 LET A$=INKEY$
1870 LOCATE 1,Y
1910 LOCATE 1,Y
2130 LOCATE 1,Y
2170 LOCATE 1,Y
2350 LOCATE 1,Y
2390 LOCATE 1,Y
2670 LOCATE 1,Y
2710 LOCATE 1,Y
3030 LET A$=INKEY$: IF A$="" THEN GOTO
     3030
3110 OPENOUT
3120 PRINT #9, I; IE; IL; IM; IO; IJ; IC
3140 PRINT #9, S$(K)
3170 PRINT #9, CL(K); IN(K)
3200 PRINT #9, E$(K)
3230 PRINT #9, V1(K)
3260 PRINT #9, E1(K)
3290 PRINT #9, HB(K); FZ(K); V2(3*K-2);
     V2(3*K-1); V2(3+K)
3320 PRINT #9, 0$(K)
3350 PRINT #9, PR(K)
3380 PRINT #9, 01(K); V3(K)
3410 PRINT #9, CN(K); V4(2*K-1); V4(2*K)
3430 CLOSEOUT
3520 LET A$=INKEY$: IF A$="" THEN GOTO
     3520
```

MSX:

```
1870 LOCATE 1,Y
1910 LOCATE 1,Y
2130 LOCATE 1,Y
2170 LOCATE 1,Y
2350 LOCATE 1,Y
2390 LOCATE 1,Y
2670 LOCATE 1,Y
2710 LOCATE 1,Y
```

A continuación vamos a ver una mini-aventura de las que se pueden realizar con este programa. Pueden confeccio-

narse aventuras todo lo complicado que se desee. Sólo tenemos que tener en cuenta la cantidad de memoria que gastamos.

La figura 1 nos muestra cómo será el plano de la casa donde se desarrolla nuestra aventura. También veremos en dicha figura cómo están numeradas las habitaciones y qué tipo de habitación es:

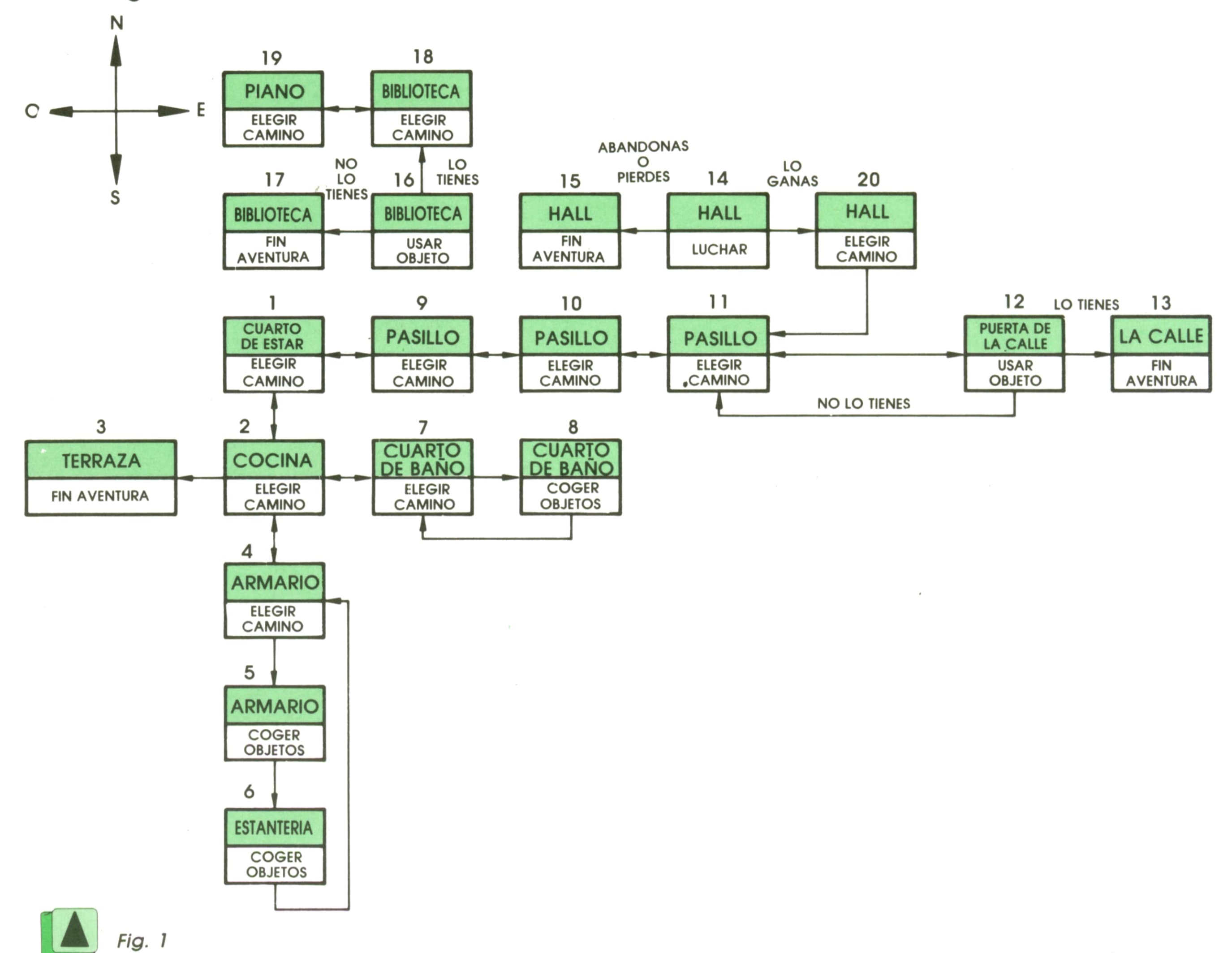


Fig. 1

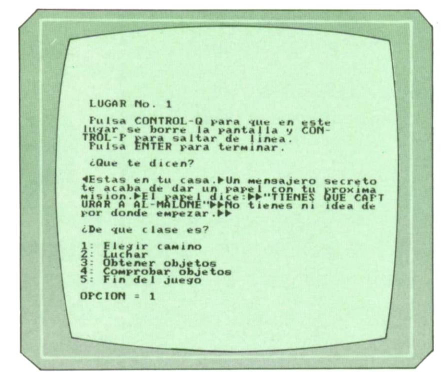
En el plano de la aventura, las flechas indican la dirección por donde puede ir el jugador. Las flechas de doble sentido indican que se pueden pasar de una habitación a otra, y viceversa. Las de una sola dirección aparecen después de pelear con alguien o después de utilizar (o no utilizar, si no se tiene) un objeto.

Hemos hecho un segundo programa que se encarga de grabar esta aventura en disco o cinta. Preferimos hacerlo así a tener que decirle al lector cómo tiene que diseñar la aventura. De todas formas, ejecutando el programa que viene más adelante y echándole un ojo al plano de la aventura se puede ver en seguida cómo se ha construido. Se recomienda al lector que haga sus propias pruebas.

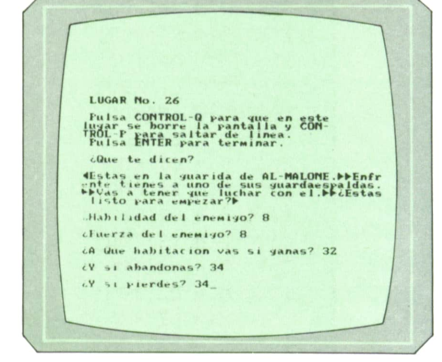
```
100 OPEN "AVENTURA" FOR OUTPUT AS #1
110 RESTORE
120 READ I,IE,IE1,IM,IO,IO1,IC
130 PRINT#1,I;IE;IE1;IM;IO;IO1;IC
140 FOR K=1 TO I
150    READ A$
160    PRINT #1,A$
170 NEXT K
```

```
180 FOR K=1 TO I
190    READ A,B
200    PRINT #1,A;B
210 NEXT K
220 FOR K=1 TO IE
230    READ A$
540 CLOSE#1
240    PRINT #1,A$
250 NEXT K
260 FOR K=1 TO IE
270    READ A
280    PRINT #1,A
290 NEXT K
300 FOR K=1 TO IE1
310    READ A
320    PRINT #1,A
330 NEXT K
340 FOR K=1 TO IM
350    READ A,B,C,D,E
360    PRINT #1,A;B;C;D;E
370 NEXT K
380 FOR K=1 TO IO
390    READ A#
400    PRINT #1,A$
410 NEXT K
420 FOR K=1 TO IO
430    READ A
440    PRINT #1,A
450 NEXT K
460 FOR K=1 TO IO1
470    READ A,B
480    PRINT #1,A;B
490 NEXT K
500 FOR K=1 TO IC
510    READ A,B,C
520    PRINT #1,A;B;C
530 NEXT K
550 DATA  20 , 21 , 9 , 1 , 4 , 4 , 2
560 DATA "Estas en el cuarto de estar de AL-MALONEQuieres escapar y no sabes com
o hacerlo."
561 DATA "Has entrado en la cocina.Enfrente ves un armario y a la derecha  la te
rraza."
562 DATA "Has entrado en la terraza y, antes de  que te des cuenta, te han disp
arado un  tiro entre los ojos.Estas muerto."
563 DATA "Te acercas al armario y ves que esta ce-rrado.No saber que hacer."
564 DATA "Has roto el armario (haciendo mucho rui-do) y ves que dentro hay comid
a."
565 DATA "Al registrar el armario ves un bote de  laca de mujer en spray."
566 DATA "Estas en el cuarto de bano.La ducha es ta lista."
567 DATA "Al mirar en la ducha alguien te agarra  del cuello.Le das una patada y
 le dejastieso.Le quitas la pistola."
568 DATA "Estas en un pasillo oscuro."
569 DATA "Estas en un pasillo oscuro."
570 DATA "Estas en un pasillo oscuro."
571 DATA "Ante ti ves la puerta de la calle.Ya te ves libre."
572 DATA "Has conseguido salir de la casa.Eres libre.FELICIDADES."
573 DATA "Entras en el hall.Te encuentras de bruces con AL-MALONE.TIENES QUE LUC
HAR !!"
574 DATA "Lo siento.AL-MALONE te ha matado.Otro dia sera."
575 DATA "Estas en la biblioteca.No ves nada.De pronto alguien te ataca."
576 DATA "Lo siento el esbirro de AL-MALONE te ha hecho pure.Estas muerto."
577 DATA "Has rociado al ganster con el Spray.Al dejarle ciego has aprobechado p
ara   partirle la cara."
578 DATA "Estas frente al piano.Es muy bonito pero no puedes tocarlo ahora."
579 DATA "Le quitas a AL-MALONE las llaves de la  casa.Ya puedes salir (si quier
es)."
580 DATA  1 , 1 , 1 , 2 , 5 , 0 , 1 , 3 , 3 , 1 , 3 , 2 , 1 , 4 , 3 , 3 , 1 , 5
 , 1 , 6 , 1 , 7 , 4 , 1 , 5 , 0 , 2 , 1 , 5 , 0 , 4 , 2 , 5 , 0 , 1 , 8 , 1 , 9
 , 3 , 4
```

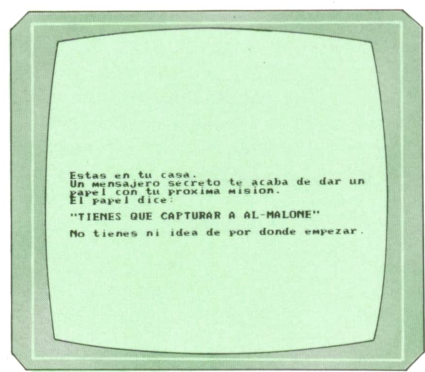
```
581 DATA "Sur"
582 DATA "Este"
583 DATA "Ir al armario"
584 DATA "Terraza"
585 DATA "Este"
586 DATA "Norte"
587 DATA "Romper el armario"
588 DATA "Pasar de hacer ruido"
589 DATA "Oeste"
590 DATA "Ducha"
591 DATA "Oeste"
592 DATA "Este"
593 DATA "Norte"
594 DATA "Oeste"
595 DATA "Este"
596 DATA "Oeste"
597 DATA "Este"
598 DATA "Norte"
599 DATA "Salir al pasillo"
600 DATA "Mirar el piano"
601 DATA "Este"
602 DATA  2 , 9 , 4 , 3 , 7 , 1 , 5 , 2 , 2 , 8 , 1 , 10 , 16 , 9 , 11 , 10 ,
 12 , 14 , 9 , 19 , 18
603 DATA  2 , 4 , 2 , 2 , 3 , 2 , 3 , 2 , 1
604 DATA  13 , 13 , 20 , 15 , 15
605 DATA "Comida"
606 DATA "Spray"
607 DATA "Pistola"
608 DATA "Llaves"
609 DATA -1 ,-1 , 0 , 0
610 DATA  1 , 6 , 1 , 4 , 1 , 7 , 1 , 11
611 DATA  4 , 13 , 11 , 2 , 18 , 17
```



Ejemplo de diseño de una aventura.



Así aparecerá el texto que se introdujo en la Fig. 2, cuando ejecutemos el programa "GAME-2".





Definición de los enemigos.

Las modificaciones que hay que hacer para que este segundo programa funcione en el AMSTRAD son las siguientes:

AMSTRAD:

```
100 OPENOUT
130 PRINT #9,I;IE;IE1;IM;IO;IO1;IC
160 PRINT #9,A$
200 PRINT #9,A;B
230 PRINT #9,A$
240 PRINT #9,A$
280 PRINT #9,A
320 PRINT #9,A
360 PRINT #9,A;B;C;D;E
400 PRINT #9,A$
440 PRINT #9,A
480 PRINT #9,A;B
520 PRINT #9,A;B;C
540 CLOSEOUT
```

El siguiente programa (GAME-2) será el que nos permita ejecutar el programa que hemos definido con GAME-1. No hace falta explicar nada sobre su funcionamiento, ya que es autoexplicativo.

```
1000 REM **************************************
1010 REM * PROGRAMA PARA EJECUTAR LOS JUEGOS *
1020 REM * REALIZADOS CON EL DISENADOR DE A- *
1030 REM * VENTURAS (GAME-1)                 *
1040 REM **************************************
1050 REM
1060 REM ********************
1070 REM *     AUTORES      *
1080 REM *     -------      *
1090 REM * MANUEL  ALFONSECA *
1100 REM *        Y         *
1110 REM * FRANCISCO MORALES *
1120 REM ********************
1130 REM
1140 REM ********************************
1150 REM **** (c) Ed. Siglo Cultural ****
1160 REM **** (c) 1987               ****
1170 REM ********************************
1180 REM
1190 REM ****************************
1200 REM * DEFINICION DE LAS TABLAS *
1210 REM ****************************
1220 REM
1230 DIM S$(300)
1240 DIM E$(800)
1250 DIM O$(100)
1260 DIM CL(300)
1270 DIM IN(300)
1280 DIM E1(300)
1290 DIM O1(300)
1300 DIM V1(800)
1310 DIM V2(250)
1320 DIM V3(100)
1330 DIM V4(170)
1340 DIM HB(100)
1350 DIM FZ(100)
1360 DIM PR(100)
1370 DIM CN(100)
1380 DIM E2(401)
1390 DIM O2(401)
1400 DIM TI(100)
1410 DIM NO(30)
1420 REM
1430 REM *** INICIALIZACION DE VARIABLES ***
1440 REM
1450 LET S1=0
1460 LET S2=0
1470 LET S3=0
1480 REM
1490 REM *** CARGA DEL JUEGO DESDE DISCO O CINTA ***
1500 REM
```

```
1510 CLS
1520 PRINT " Comprueba que la cinta o el"
1530 PRINT "disco estan colocados."
1540 PRINT
1550 PRINT "PULSA UNA TECLA"
1560 LET A$=INPUT$(1)
1570 PRINT
1580 INPUT "(Nombre de la aventura";F$
1590 IF F$="" THEN GOTO 4050
1600 OPEN F$ FOR INPUT AS #1
1610 PRINT
1620 PRINT "CARGANDO ... ";F$
1630 PRINT
1640 PRINT "Espera un momento."
1650 PRINT
1660 INPUT #1,I,IE,IE1,IM,IO,IO1,IC
1670 FOR K=1 TO I
1680    LINE INPUT #1,S$(K)
1690 NEXT K
1700 FOR K=1 TO I
1710    INPUT #1,CL(K),IN(K)
1720 NEXT K
1730 FOR K=1 TO IE
1740    LINE INPUT #1,E$(K)
1750 NEXT K
1760 FOR K=1 TO IE
1770    INPUT #1,V1(K)
1780 NEXT K
1790 FOR K=1 TO IE1
1800    INPUT #1,E1(K)
1810 NEXT K
1820 FOR K=1 TO IM
1830    INPUT #1,HB(K),FZ(K),V2(3*K-2),V2(3*K-1),V2(3*K)
1840 NEXT K
1850 FOR K=1 TO IO
1860    LINE INPUT #1,O$(K)
1870 NEXT K
1880 FOR K=1 TO IO
1890    INPUT #1,PR(K)
1900 NEXT K
1910 FOR K=1 TO IO1
1920    INPUT #1,O1(K),V3(K)
1930 NEXT K
1940 FOR K=1 TO IC
1950    INPUT #1,CN(K),V4(2*K-1),V4(2*K)
1960 NEXT K
1970 CLOSE #1
1980 PRINT "La aventura: ";CHR$(34);F$;CHR$(34);" esta cargada."
1990 PRINT
2000 PRINT "PULSA UNA TECLA"
2010 LET A$=INPUT$(1)
2020 CLS
2030 REM
2040 REM *** COMIENZA EL JUEGO ***
2050 REM
2060 LET PO=1
2070 FOR J=1 TO IE1
2080    FOR K=1 TO J
2090       LET E2(J+1)=E2(J)+E1(K)
2100    NEXT K
2110 NEXT J
2120 FOR J=1 TO IO1
2130    FOR K=1 TO J
2140       LET O2(J+1)=O2(J)+O1(K)
2150    NEXT K
2160 NEXT J
2170 LET IT=0
2180 RANDOMIZE TIME
2190 LET HA=10+INT(RND*10)
```

```
2200 LET FU=10+INT(RND*10)
2210 LET DO=200+INT(RND*100)
2220 PRINT " Aqui estan tus poderes:"
2230 PRINT
2240 PRINT "Tu habilidad es ....";HA
2250 PRINT "Tu fuerza es .......";FU
2260 PRINT
2270 PRINT "Tienes";DO;"Pesetas"
2280 PRINT
2290 PRINT "PULSA UNA TECLA"
2300 LET A$=INPUT$(1)
2310 CLS
2320 REM
2330 REM *** BUCLE PRINCIPAL DE JUEGO ***
2340 REM
2350 A$=S$(PO)
2360 PRINT
2370 FOR Z=1 TO LEN(A$)
2380    IF MID$(A$,Z,1)=CHR$(16) THEN PRINT CHR$(13);:GOTO 2410
2390    IF MID$(A$,Z,1)=CHR$(17) AND Z=1 THEN CLS:GOTO 2410
2400    PRINT MID$(A$,Z,1);
2410 NEXT Z
2420 PRINT
2430 PRINT
2440 LET A=IN(PO)
2450 ON CL(PO) GOTO 2470,2640,3060,3920,4040
2460 REM
2470 REM *** ELEGIR CAMINO ***
2480 REM
2490 LET B=1
2500 IF E1(A)=1 THEN GOTO 2610
2510 FOR K=1 TO E1(A)
2520    PRINT K;":";E$(E2(A)+K)
2530 NEXT K
2540 PRINT
2550 PRINT "(Que camino eliges? ";CHR$(219);CHR$(29);
2560 LET A$=INPUT$(1)
2570 IF A$<"1" OR A$>"9" OR VAL(A$)>E1(A) THEN GOTO 2560
2580 LET B=VAL(A$)
2590 PRINT A$
2600 PRINT
2610 LET PO=V1(E2(A)+B)
2620 GOTO 2330
2630 REM
2640 REM *** LUCHA CONTRA UN ENEMIGO ***
2650 REM
2660 LET H=HB(A)
2670 LET F=FZ(A)
2680 IF FU>4 THEN GOTO 2760
2690 PRINT
2700 PRINT " Te estas agotando de luchar."
2710 PRINT " (Quieres abandonar? (S/N) ";CHR$(219);CHR$(29);
2720 LET A$=INPUT$(1)
2730 IF A$<>"S" AND A$<>"s" AND A$<>"N" AND A$<>"n" THEN GOTO 2720
2740 PRINT A$
2750 IF A$="S" OR A$="s" THEN GOTO 3030
2760 LET AA=HA+1+INT(RND*11)
2770 LET BB=H+1+INT(RND*11)
2780 PRINT
2790 IF AA=BB THEN PRINT " Los dos fallasteis el golpe."
2800 IF AA>BB THEN PRINT " Le has dado y le has dejado tonto."
2810 IF AA<BB THEN PRINT " Te ha dado un punetazo.":PRINT "Te has quedado atonta
do y":PRINT "pierdes un punto de tu fuerza."
2820 PRINT
2830 IF AA<BB THEN FU=FU-1
2840 FOR V=1 TO 400: NEXT V
2850 IF FU=0 THEN GOTO 2970
2860 IF AA>BB THEN F=F-1
2870 IF F=0 THEN GOTO 2900
2880 GOTO 2680
```

```
2890 PRINT
2900 PRINT " Le has matado. Eres un fiera."
2910 PRINT " Enhorabuena campeon."
2920 PRINT
2930 LET PO=V2(3*A-2)
2940 FOR V=1 TO 1000:NEXT V
2950 GOTO 2330
2960 PRINT
2970 PRINT " Te ha vencido. Lo siento"
2980 PRINT " Mucha sera la gente que te llore."
2990 FOR V=1 TO 1000:NEXT V
3000 PRINT
3010 LET PO=V2(3*A)
3020 GOTO 2330
3030 LET PO=V2(3*A-1)
3040 GOTO 2330
3050 REM
3060 REM *** OBTENER OBJETOS ***
3070 REM
3080 LET Q=0
3090 FOR K=1 TO O1(A)
3100    IF PR(O2(A)+K)<>0 THEN S1=1:GOTO 3200
3110    IF Q=0 THEN PRINT:PRINT " Te dan los siguientes objetos:"
3120    LET Q=1
3130    PRINT
3140    PRINT O$(O2(A)+K)
3150    LET A$=O$(O2(A)+K)
3160    GOSUB 4160
3170    LET TI(IT+1)=O2(A)+K
3180    LET IT=IT+1
3190    LET O1(A)=O1(A)-1
3200 NEXT K
3210 LET Q=0
3220 FOR K=1 TO O1(A)
3230    IF PR(O2(A)+K)<=0 THEN S2=1:GOTO 3290
3240    IF Q=0 THEN PRINT:PRINT "Puedes comprar los siguientes objetos"
3250    IF Q=0 THEN PRINT:PRINT "con las";DO;"pesetas que tienes:":PRINT
3260    LET Q=Q+1
3270    PRINT Q;":";O$(O2(A)+K)
3280    LET NO(Q)=O2(A)+K
3290 NEXT K
3300 IF Q=0 THEN GOTO 3600
3310 PRINT:PRINT "(Cuantos quieres? ";CHR$(219);CHR$(29);
3320 LET A$=INPUT$(1)
3330 IF A$<"0" OR A$>"9" THEN GOTO 3320
3340 LET B=VAL(A$)
3350 PRINT A$
3360 PRINT
3370 IF B>O1(A) THEN PRINT:PRINT " No hay tantos objetos.":GOTO 3310
3380 IF B=0 GOTO 3590
3390 PRINT "(Que numeros quieres?"
3400 PRINT "Damelos de uno en uno."
3410 PRINT
3420 FOR K=1 TO B
3430    INPUT "objeto = ";N
3440    IF N<1 OR N>O1(A) THEN PRINT:PRINT " Ese objeto no esta.":PRINT:GOTO 343
0
3450    LET N=NO(N)
3460    IF PR(N)>DO THEN GOTO 3530
3470    LET TI(IT+1)=N
3480    LET IT=IT+1
3490    LET DO=DO-PR(N)
3500    LET A$=O$(O2(A)+N)
3510    GOSUB 4160
3520    GOTO 3570
3530    PRINT:PRINT "No tienes bastante dinero para"
3540    PRINT "comprar ";O$(O2(A)+N)
3550    LET K=O1(A)
3560    GOTO 3580
```

```
3570    LET O1(A)=O1(A)-1
3580 NEXT K
3590 LET Q=0
3600 FOR K=1 TO O1(A)
3610    IF PR(O2(A)+K)>=0 THEN S3=1:GOTO 3660
3620    IF Q=0 THEN PRINT "Puedes coger los siguientes objetos:":PRINT
3630    LET Q=Q+1
3640    PRINT Q;":";O$(O2(A)+K)
3650    LET NO(Q)=O2(A)+K
3660 NEXT K
3670 IF Q=0 THEN GOTO 3870
3680 PRINT:PRINT "(Cuantos quieres? ";CHR$(219);CHR$(29);
3690 LET A$=INPUT$(1)
3700 IF A$<"0" OR A$>"9" THEN GOTO 3690
3710 LET B=VAL(A$)
3720 PRINT A$
3730 IF B>O1(A) THEN PRINT:PRINT " No hay tantos objetos.":GOTO 3680
3740 IF B=0 GOTO 3870
3750 PRINT:PRINT "(Que numeros quieres?"
3760 PRINT "Damelos de uno en uno.":PRINT
3770 FOR K=1 TO B
3780    INPUT "Objeto = ";N
3790    IF N<1 OR N>O1(A) THEN PRINT:PRINT " Ese objeto no esta.":PRINT:GOTO 378
0
3800    LET A$=O$(O2(A)+N)
3810    GOSUB 4160
3820    LET O1(A)=O1(A)-1
3830    LET N=NO(N)
3840    LET TI(IT+1)=N
3850    LET IT=IT+1
3860 NEXT K
3870 LET PO=V3(A)
3880 LET S1=0:LET S2=0:LET S3=0
3890 FOR V=1 TO 500:NEXT V
3900 GOTO 2330
3910 REM
3920 REM *** COMPROBAR OBJETOS ***
3930 REM
3940 FOR K=1 TO IT
3950    IF CN(A)=TI(K) THEN GOTO 4000
3960 NEXT K
3970 PRINT:PRINT "Lo siento. No tienes ";O$(CN(A))
3980 LET PO=V4(2*A)
3990 GOTO 2330
4000 PRINT:PRINT "Tienes suerte. Tenias ";O$(CN(A))
4010 LET PO=V4(2*A-1)
4020 FOR V=1 TO 500:NEXT V
4030 GOTO 2330
4040 REM CASILLA FINAL DEL JUEGO
4050 PRINT
4060 PRINT " El juego ha terminado."
4070 PRINT
4080 PRINT " (Quieres jugar otra vez? (S/N)"
4090 LET A$=INPUT$(1)
4100 IF A$="s" OR A$="S" THEN CLS:GOTO 1600
4110 IF A$<>"n" AND A$<>"N" THEN GOTO 4090
4120 PRINT
4130 PRINT " Hasta la vista."
4140 PRINT:PRINT:PRINT
4150 END
4160 REM
4170 REM *********************************
4180 REM * RECONOCER OBJETOS ESPECIALES *
4190 REM *********************************
4200 REM
4210    IF A$="Comida" THEN LET FU=FU*2:PRINT:PRINT " Ahora tu fuerza es:";FU:PR
INT
4220    IF A$="Daga" THEN LET HA=HA+10:PRINT:PRINT " Ahora tu habilidad es:"HA:P
RINT
```

```
4230    IF A$="Sable" THEN LET HA=HA*2:PRINT:PRINT " Tu habilidad ha aumentado a
:"HA:PRINT
4240    IF A$="Pistola" THEN LET HA=HA*3:PRINT:PRINT " Tienes una habilidad de:"
HA:PRINT
4250    IF A$="Leche" THEN LET FU=INT(FU*1.5):PRINT:PRINT " Tienes una fuerza de
:";FU:PRINT
4260    IF A$="Vitaminas" THEN LET FU=FU*3):PRINT:PRINT " Tu fuerza es igual a:"
;FU:PRINT
4270 RETURN
```

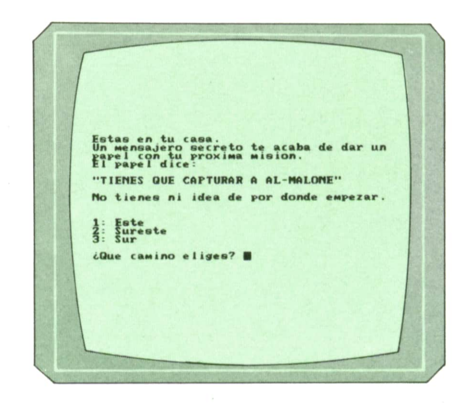
Las modificaciones que hay que realizar son las siguientes:

AMSTRAD:

```
1560 LET A$=INKEY$: IF A$="" THEN GOTO
     1560
1600 OPENIN
1660 INPUT #9,I,IE,IE1,IM,I0,I01,IC
1680 LINE INPUT #9,S$(K)
1710 INPUT #9,CL(K),IN(K)
1740 LINE INPUT #9,E$(K)
1770 INPUT #9,V1(K)
1800 INPUT #9,E1(K)
1830 INPUT #9,HB(K), FZ(K), V2(3*K-2),
     V2(3*K-1), V2(3*K)
1860 LINE INPUT #9,0$(K)
1890 INPUT #9,PR(K)
1920 INPUT #9,O1(K),V3(K)
1950 INPUT #9,CN(K),V4(2*K-1),V4(2*K)
1970 CLOSEOUT
2010 LET A$=INKEY$:IF A$="" THEN GOTO
     2010
2190 LET HA=10+INT (RND(1)*10)
2200 LET FU=10+INT (RND(1)*10)
2210 LET DO=200+INT (RND(1)*100)
2300 LET A$=INKEY$:IF A$"" THEN GOTO
     2300
2560 LET A$=INKEY$:IF A$="" THEN GOTO
     2560
2720 LET A$=INKEY$:IF A$"" THEN GOTO
     2720
2760 LET AA=HA+1+INT (RND(1)*11)
2770 LET BB=H+1+INT (RND(1)*11)
3320 LET A$=INKEY$:IF A$="" THEN GOTO
     3320
3690 LET A$=INKEY$:IF A$="" THEN GOTO
     3690
4090 LET A$=INKEY$:IF A$="" THEN GOTO
     4090
```

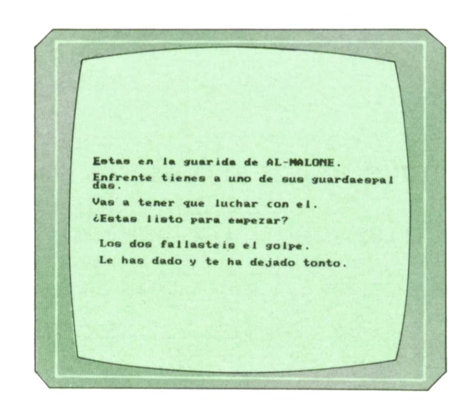
MSX:

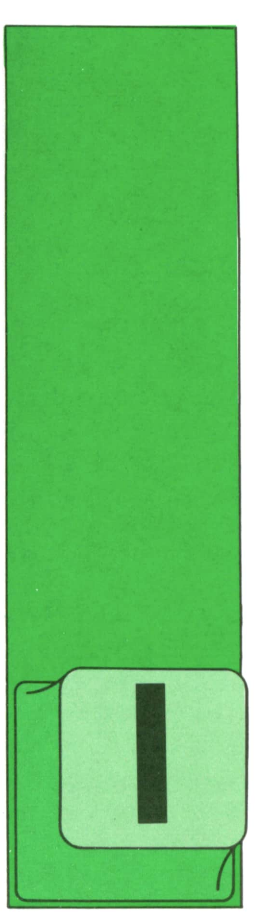
```
2190 LET HA=10+INT (RND(1)*10)
2200 LET FU=10+INT (RND(1)*10)
2210 LET DO=200+INT (RND(1)*100)
2760 LET AA=HA+1+INT (RND(1)*11)
2770 LET BB=H+1+INT (RND(1)*11)
```





Ejecutando el programa "GAME-2".





TECNICAS DE ANALISIS

DISEÑO DE DOCUMENTOS

INDEPENDIENTEMENTE del hecho de que el documento sea preimpreso o se diseñe para ser escrito íntegramente con la impresora del ordenador, hay que tener en cuenta otros elementos respecto de su aspecto general:

b) Presentación. Aunque, evidentemente, siempre es útil cuidar que el aspecto de cualquier documento sea agradable, éste es un detalle muy a tener en cuenta cuando los documentos que se diseñan van a ser utilizados por personas ajenas al proceso de datos (e, incluso en ocasiones, con cierta animadversión hacia él). Así, sin despreciar los restantes detalles, hay que cuidar algunas características que dan un aspecto más agradable al documento que se diseña:

— *Tamaño de la letra.* En la mayoría de las impresoras la altura de los caracteres es fija, pero no la anchura de ellos: conviene disponer que se escriba con la letra mayor que el contenido del documento, su tamaño exterior, etc., permitan. Aunque en las impresoras de matriz de puntos (las más extendidas en la informática personal y profesional) se puede variar dentro de un documento (e incluso dentro de una línea) la anchura de los caracteres, en las impresoras de líneas esto no es posible. En cualquier caso, debe estudiarse con cuidado esta variación de la anchura de los caracteres, pues suele ser fuente de errores y disminuye la "portabilidad" del programa que escribe los datos (pueden variar los caracteres especiales de control de la impresora al pasar a otro equipo).

También hay que estudiar el valor óptimo; de la separación entre líneas (1/6, 1/8 de pulgada o valores no estándar), pues es un parámetro variable en prácticamente todas las impresoras profesionales.

Por el contrario, hay quien defiende la uniformidad de modo de escritura (tamaño de letra, separación de líneas...) por razones de ahorro de tiempo, simplicidad, transportabilidad, etc. En los centros de proceso de datos (CPD's) es usual que se establezcan valores estándares para estas características (variables, a lo sumo, según el tipo de documento, pero nunca al arbitrio del analista).

— *Color y sombreados.* Aunque el papel no sea preimpreso, se puede elegir, en ocasiones, el color de fondo (especialmente si se escribe en hojas sueltas) o, al menos, entre un papel pautado (el conocido "papel pijama") o uno blanco.

— *Gráficos.* Siempre que sea posible se deberá incluir algún pequeño gráfico, algún rótulo fijo (en letra grande, subrayado, todo en mayúsculas...), algunas líneas separadoras, etc., para dar un poco de variedad y "movimiento" al documento. También es útil con esta finalidad sustituir (en la medida de lo posible) las tablas y cuadros de cifras por gráficos o re-

presentaciones visuales de cualquier tipo.

— *Márgenes.* Hay que tener en cuenta que un margen izquierdo ancho "aligera" un documento. Una buena práctica adicional es no aprovechar el espacio de papel hasta el borde derecho, sino dejar un buen margen y "sacar" a este margen algunas informaciones y datos que interese subrayar.

c) Distribución de los datos en el documento. Es importante que la distribución de las informaciones que se ofrecen en el documento de salida sea:

— *Clara.* Es decir, que el documento tenga cierta lógica, se agrupen en él los datos por conceptos y "separen" en alguna medida las distintas "zonas" del documento.

— *Ordenada.* En efecto, conviene que exista cierta secuencia en los datos que se presentan: datos de total al principio, con las líneas de desglose detrás o líneas de detalle con sus totales acumulados al final.

— *Condensada.* Deben escribirse todos los datos necesarios, pero teniendo en cuenta que es tan desagradable para quien utiliza un listado no tener el dato que necesita como que esa información esté rodeada de otras muchas cifras no significativas, entre las cuales "se pierde".

d) Tamaño del documento de salida. En la mayoría de los CPD's existen normas sobre los tamaños de los documentos y a ellas habrá de someterse el analista al diseñar un impreso de salida; pero aunque no sea así, es útil tratar de mantenerse dentro de los valores estándar: la longitud de los papeles más comunes es de 11 ó 12 pulgadas (27,94 ó 30,48 cm), lo que supone escribir unas 56 ó 62 líneas, respectivamente (a 1/6 pulgada y dejando 10 líneas para cabeceras, pies, numeración de páginas, etc.). En cuanto a la anchura, lo normal es disponer de 80 columnas (con un ancho unitario de 1/10 de pulgada) o 132 columnas, lo que supone un papel con una anchura total de 10 y 15 pulgadas respectivamente (variando el ancho en función del tipo de papel elegido, las características de la impresora que se utiliza, etc.).

e) Formato general del documento. En todo documento de salida se suelen incluir tres zonas de datos (y en ocasiones un grupo de comentarios, códigos, instrucciones de utilización, etc).

Estas zonas conviene que estén claramente separadas para una más fácil identificación de los datos escritos. Suele existir una zona de cabeceras, otra de pies de página y otra tercera que forma el cuerpo del documento.

— *Cabeceras.* Es usual disponer varias líneas (de una a tres habitualmente) con los datos fijos del documento: nombre del documento, identificación del organismo, empresa o departamento que emite esos datos, fecha de creación, códigos identificativos, número de página, etcétera. Es importante determinar si las diferentes hojas del documento van a ser separadas (para poner en cada una de ellas toda la identificación) o no (en cuyo caso se limitarán los datos incluidos en cada hoja a lo estrictamente necesario para su localización y para que resulte estético el conjunto).

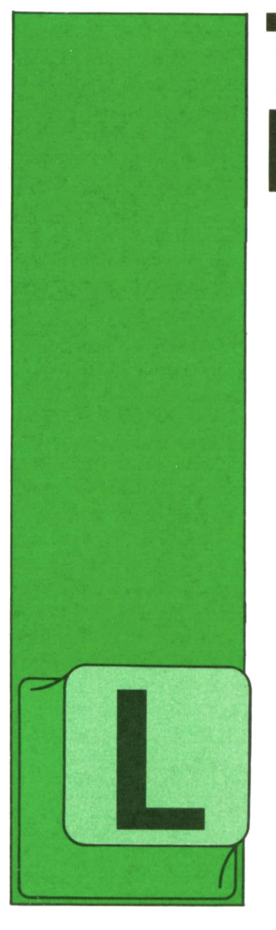
— *Pies de página.* Se pueden incluir a pie de página (una o dos líneas a lo sumo) comentarios o datos complementarios a los incluidos en el cuerpo del documento. A veces se ponen algunas de las informaciones indicadas anteriormente para las cabeceras (nombre y número de página, es lo más usual) al pie del documento. En ocasiones (casi siempre en preimpresos de una sola hoja) se incluyen al pie espacios y recuadros para controles, firmas o anotaciones manuales diversas.

— *Centro del documento.* La mayoría del espacio útil de una página debe estar dedicada a los datos variables. Es usual establecer la longitud del documento en 56 ó 62 líneas (papeles de 11 ó 12 pulgadas, respectivamente).

Naturalmente, estos apartados se ven de un modo diferente si se diseña un preimpreso o si ha de escribirse todo en la impresora del ordenador; en el primer caso, se pueden preparar datos preimpresos exhaustivos con tablas de códigos, instrucciones, etc.; en el segundo, ha de reducirse al máximo la escritura de datos fijos que consumen tiempo de proceso y espacio de memoria (en el ordenador) y tiempo de impresión. Si se utiliza un papel preimpreso se puede minimizar (o, incluso, suprimir) el pie de página y pasar todos los datos que se suelen incluir en él, al dorso del papel.

TECNICAS DE PROGRAMACION

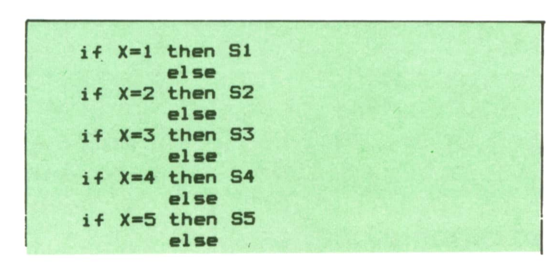
Otras formas de la instrucción condicional



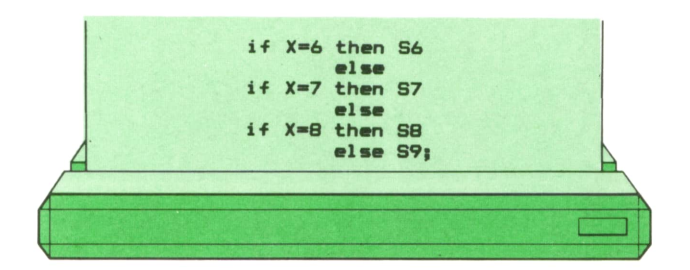
A instrucción IF-THEN-ELSE es la única forma de instrucción condicional del lenguaje BASIC. Sin embargo, otros lenguajes más avanzados permiten utilizar otras, más complejas, que aumentan las posibilidades del programador y simplifican la escritura y la interpretación de los programas. Una de estas formas es la instrucción CASE de PASCAL.

Supongamos que tenemos una variable X que puede tomar varios valores diferentes, por ejemplo: 1, 2, 3, 4, 5, 6, 7, 8. En nuestro programa deseamos ejecutar la instrucción S1 si el valor de X es igual a 1; pero si es igual a 2, en lugar de S1 queremos ejecutar S2, y así sucesivamente. Finalmente, si X es igual a 8, ejecutaremos la instrucción S8. Pero si X no tiene ninguno de los valores anteriores, ejecutaremos la instrucción S9.

Por supuesto, es posible construir un programa que se comporte así, utilizando tan sólo la instrucción IF-THEN-ELSE. Veamos cómo se haría:

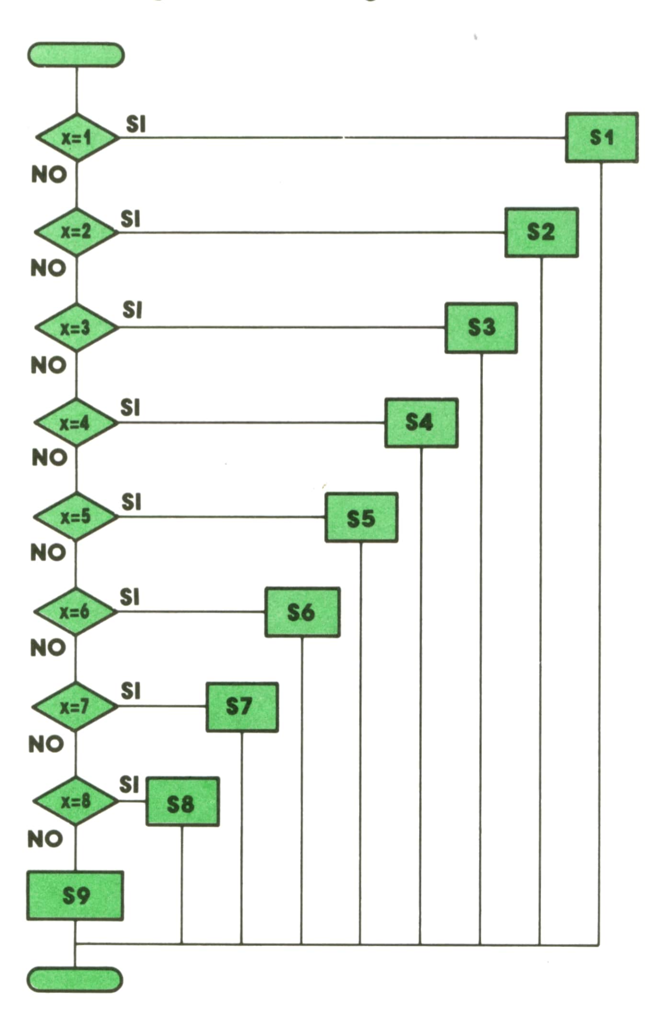


```
if X=1 then S1
        else
if X=2 then S2
        else
if X=3 then S3
        else
if X=4 then S4
        else
if X=5 then S5
        else
```

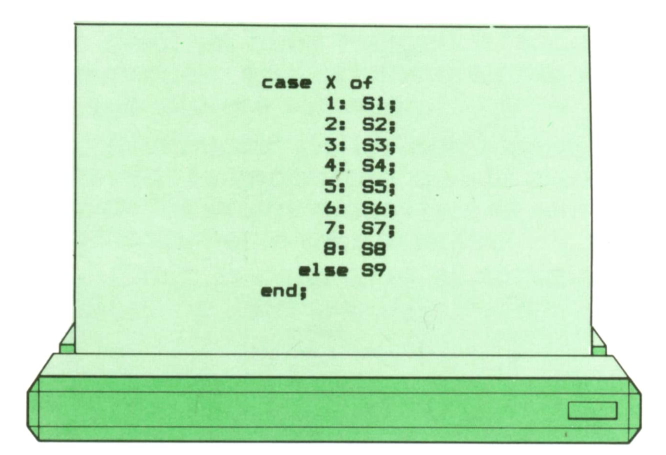


```
if X=6 then S6
        else
if X=7 then S7
        else
if X=8 then S8
        else S9;
```

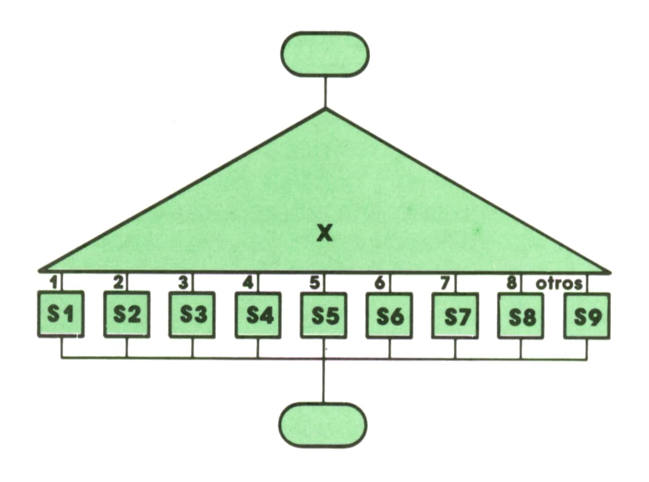
El organigrama que corresponde a este programa es el siguiente:



que, como se ve, es bastante complejo para tratarse de una sola instrucción. Por eso, en PASCAL existe la estructura CASE, que viene a ser una abreviatura de la instrucción anterior. Con ella, nuestro ejemplo se escribiría así:

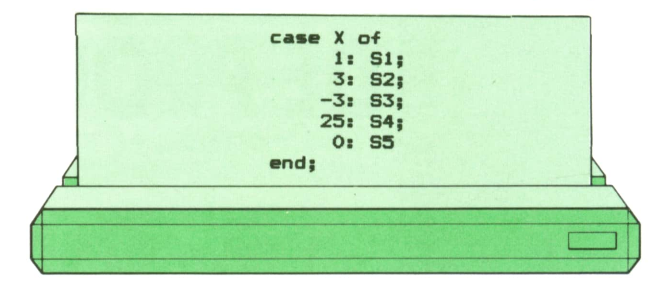


cuyo organigrama puede reducirse a:

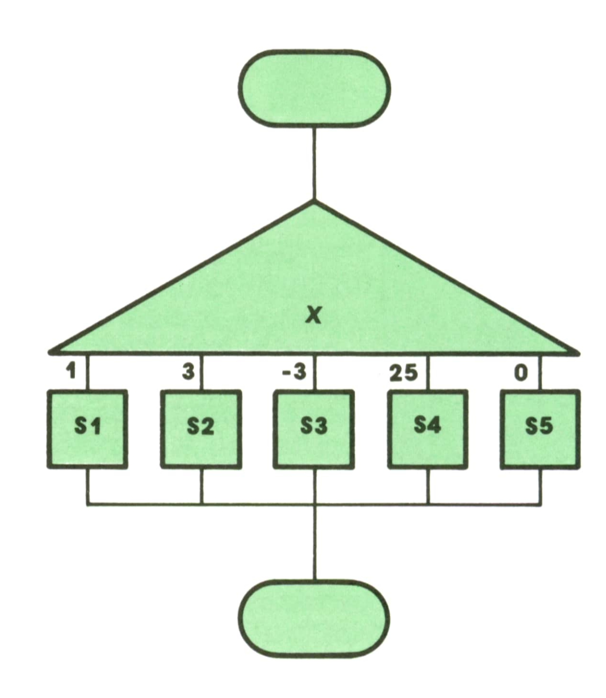


No todos los compiladores de PASCAL permiten utilizar la palabra reservada ELSE, que aparece al final de nuestro ejemplo para especificar lo que hay que hacer si el valor de X no pertenece a ninguno de los casos especificados. En algunos compiladores esta palabra se sustituye por OTHERWISE ("de lo contrario", en inglés). En otros, está totalmente prohibida. Si esto ocurre, cuando el valor de la variable de control de la estructura CASE se encuentra fuera de lo previsto, no se ejecutará nada. El organigrama correspondiente será el mismo de la figura anterior, eliminando el bloque marcado con S9.

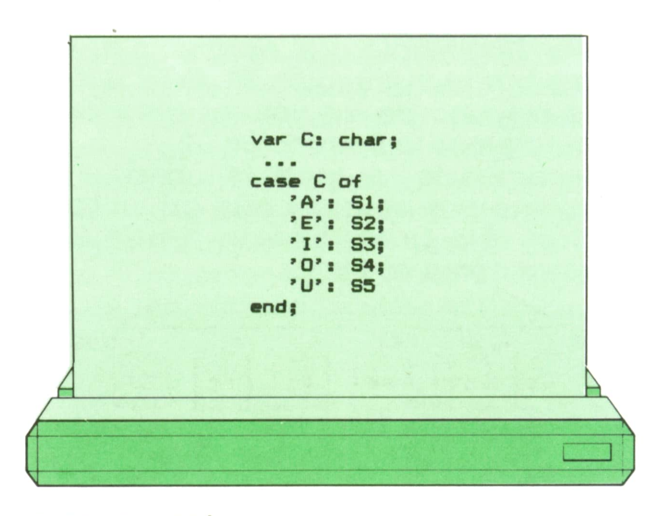
No es necesario que los valores que puede tomar la variable de control del bloque CASE tengan que ser consecutivos. Podrían estar salteados. Por ejemplo:



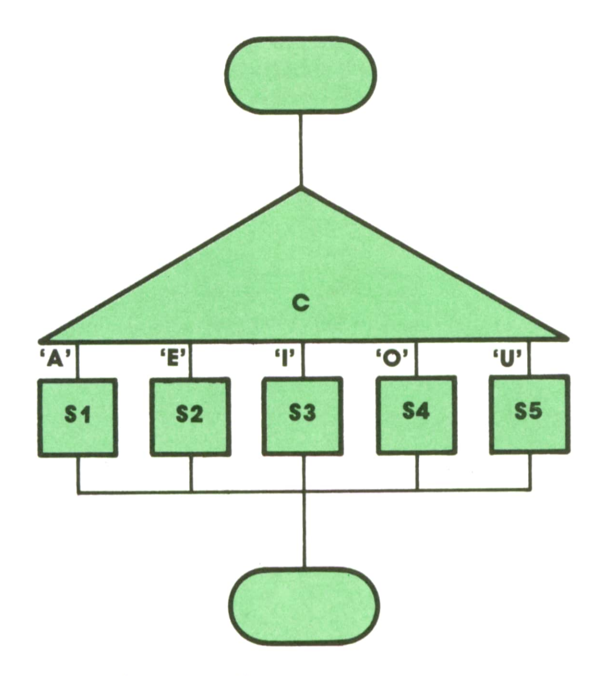
que corresponde al organigrama:



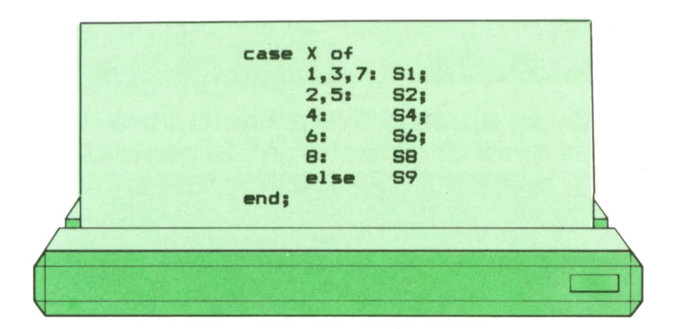
Ni siquiera es necesario que se trate de números. Podría tratarse de otros tipos de datos. Por ejemplo:



cuyo organigrama es:

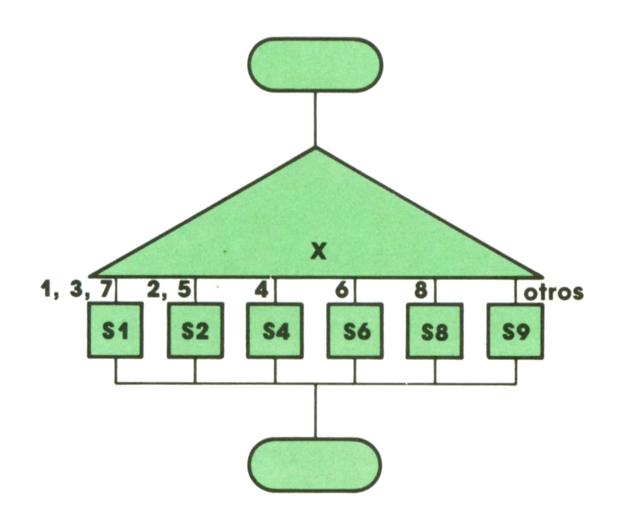


También podría ocurrir que varios de los casos coincidieran. En nuestro primer ejemplo, S1 podría ser igual a S3 y a S7, mientras S2 podría ser igual a S5, S4, S6, S8 y S9 serían independientes entre sí. En tal caso, la instrucción CASE puede simplificarse aún más, reuniendo los casos comunes en uno solo:

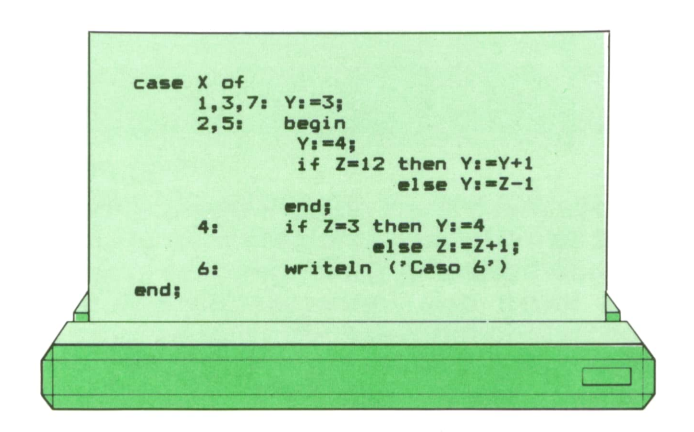


```
case X of
     1,3,7: S1;
     2,5:   S2;
     4:     S4;
     6:     S6;
     8:     S8
     else   S9
end;
```

y el organigrama sería:

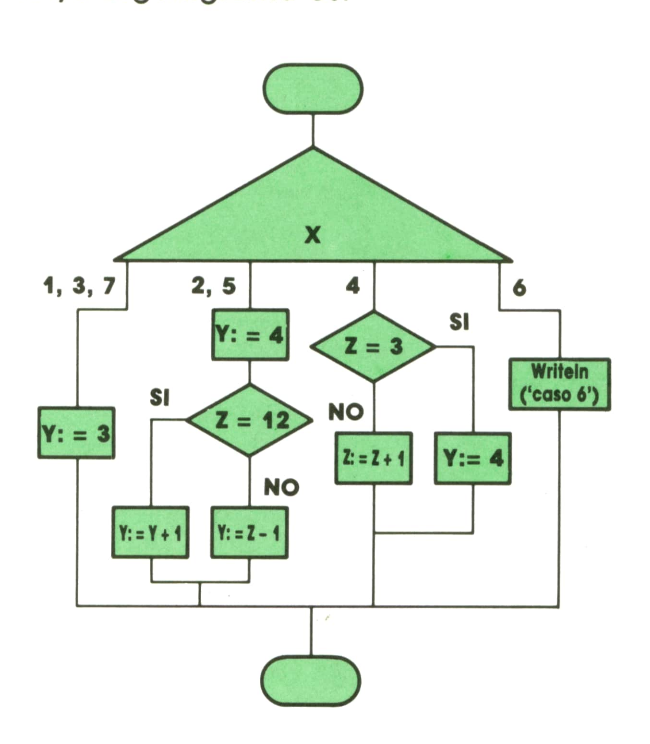


Las instrucciones S1, S2, etc., pueden ser de cualquier tipo. Podrían ser instrucciones IF-THEN-ELSE, bucles, instrucciones de asignación o bloques secuenciales de instrucciones. Veamos un ejemplo de este último caso:



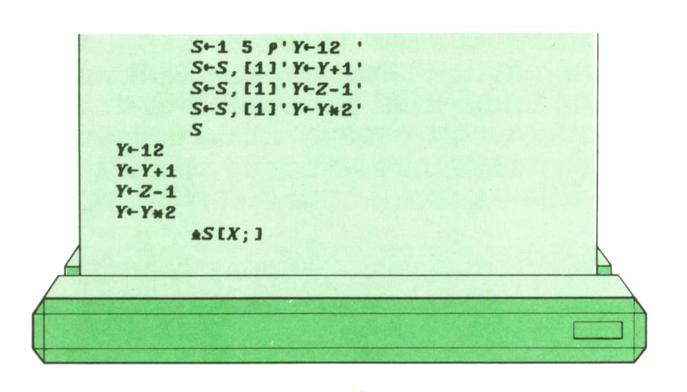
```
case X of
     1,3,7: Y:=3;
     2,5:   begin
             Y:=4;
             if Z=12 then Y:=Y+1
                     else Y:=Z-1
            end;
     4:     if Z=3 then Y:=4
                   else Z:=Z+1;
     6:     writeln ('Caso 6')
end;
```

cuyo organigrama es:



En el lenguaje APL es posible construir estructuras de control semejantes a ésta utilizando la operación descrita al hablar del bloque IF-THEN. Para ello, basta construir una matriz de caracteres, cada una de cuyas filas sea una de las instrucciones S1, S2, etc. Después indexaremos esa matriz por el número de la fila que se desea ejecutar y aplicaremos el símbolo de ejecución.

Veamos algunos ejemplos:



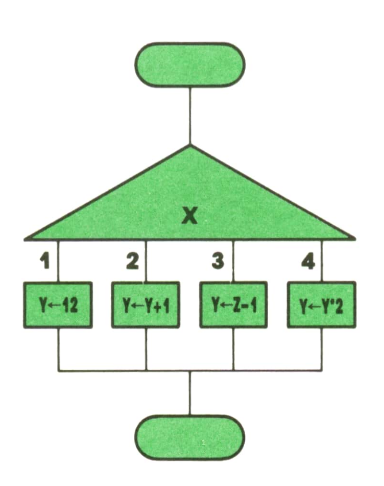
La primera línea de este ejemplo asigna a la variable S una matriz de una fila y cinco columnas (la longitud de la línea más larga que tenemos que ejecutar condicionalmente) cuyo valor es igual a la primera instrucción a ejecutar, encerrada entre comillas y completada con blancos hasta la longitud indicada.

La segunda línea añade a la variable S una nueva fila (la segunda) con la segunda instrucción. También en este caso sería preciso completarla con blancos hasta el tamaño indicado, de no ser porque esta instrucción es exactamente igual a dicho tamaño.

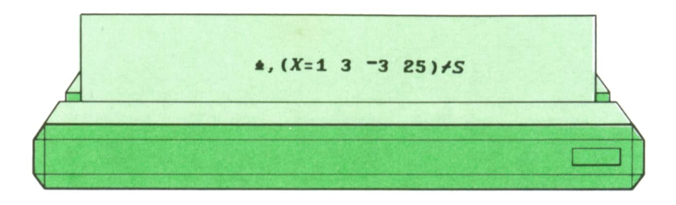
Las líneas tercera y cuarta concatenan nuevas líneas a la variable S, que así pasa a tener cuatro filas y cinco columnas.

La línea quinta nos presenta el valor de S, para comprobar que es exactamente el deseado.

Por fin, la última línea indexa la tabla S por la fila correspondiente al valor de X (es decir, selecciona una fila) y la ejecuta. Por tanto, si X es igual a 1, se ejecutará la primera fila; si es igual a 2, la segunda; si es igual a 3, la tercera y si es igual a 4, la cuarta. Veamos el organigrama correspondiente:

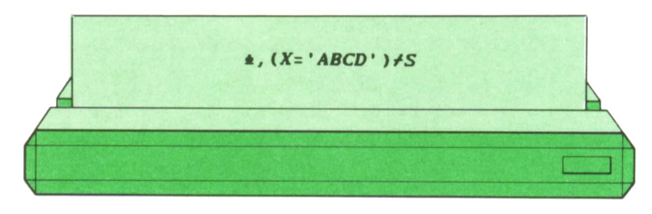


El ejemplo anterior sólo es aplicable en el caso de que los valores de X sean consecutivos, comenzando en 1. El ejemplo siguiente elimina esta restricción. Supondremos que S es la misma matriz que en el caso anterior.



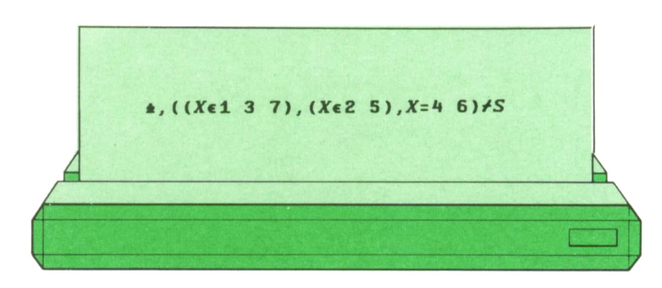
En esta instrucción, si X es igual a 1, se ejecutará la primera línea de S; si es igual a 2, la segunda; y así sucesivamente. El organigrama será el mismo que el del programa anterior, sustituyendo los valores de X (1, 2, 3, 4) por los nuevos (1,3, −3,25).

No es necesario que los valores de X sean números. Pueden muy bien ser caracteres, como en el siguiente ejemplo:

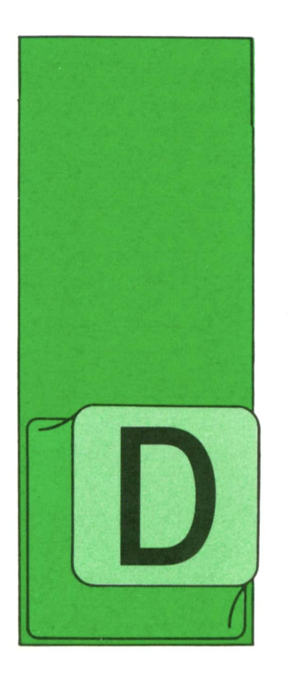


donde se ejecutará la primera línea de S si X es igual al carácter 'A', la segunda si es igual a 'B', y así sucesivamente.

También es posible agrupar varios valores de X que correspondan a la misma línea de S. Para ello, APL utiliza la terminología de la teoría de conjuntos:



En este caso, si X vale 1, 3 ó 7 (es decir, si pertenece al conjunto 1 3 7), se ejecutará la primera línea de S. Si X vale 2 ó 5 se ejecutará la segunda. Si es igual a 4 la tercera y si es igual a 6 la cuarta. Obsérvese que es posible mezclar las dos formas de seleccionar las líneas en la misma instrucción.



APLICACIONES

HOJAS DE CALCULO



LOTUS-123

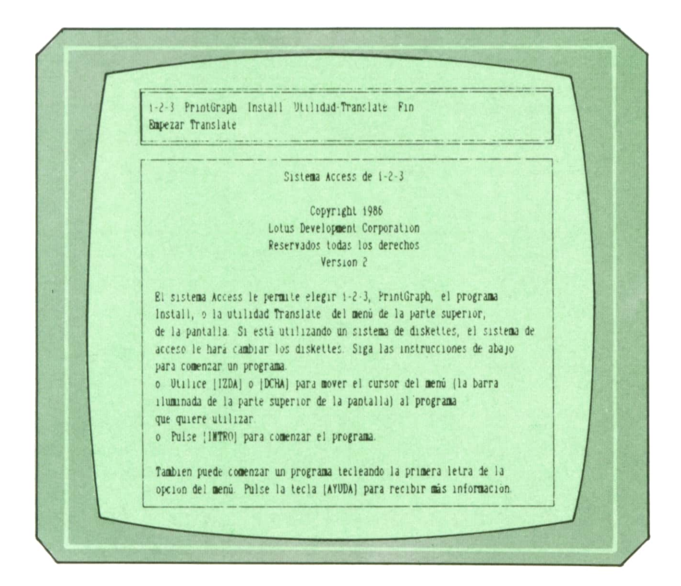
DISTRIBUIDO por Lotus Development, tenemos ante nosotros un paquete de aplicación muy completo, con una gran aceptación en el mundo profesional donde se adopta la hoja de cálculo o el paquete completo como herramienta de gestión.

Para utilizarlo debe disponer de un equipo de IBM o compatible con un mínimo de 256 Kb de memoria RAM.

En Lotus 1-2-3 la estructura fundamental para organizar y almacenar los datos es la hoja de trabajo. La capacidad de la misma es de 256 columnas y 8.192 filas.

Pero dejémonos de comentarios y pasemos a iniciar la sesión de trabajo correspondiente:

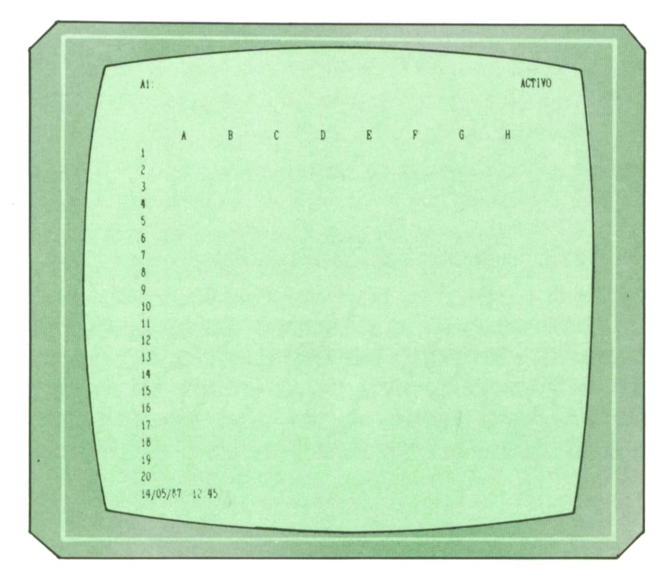
Teclee LOTUS y pulse (Intro)



Aparece en su pantalla la pantalla 1, en la que le permite seleccionar la aplicación Lotus que desee. Para ello no tiene más que moverse con las flechas de cursor, situarse sobre la opción elegida y pulsar (Intro), o bien pulse la inicial de dicha opción.

Nuestra intención es conocer la hoja de cálculo; para ello nos situamos sobre la palabra 1-2-3 del menú superior y pulsamos (Intro).

1-2-3 es un solo programa que combina tres entornos de trabajo: una hoja de cálculo, base de datos y gráficos.



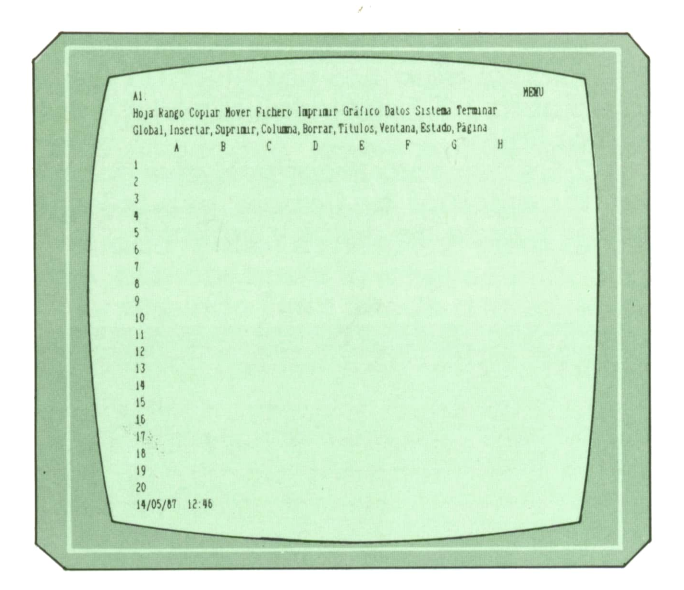
La hoja de trabajo aparece ante nosotros. En la parte superior de la pantalla figura el nombre de la celda activa; si se desplaza con las flechas de cursor verá cómo varía este nombre para dar el correspondiente a la posición del cursor. Seguido a los dos puntos, en el espacio en blanco, aparecerá el contenido de la celda. Por ejemplo: teclee con el cursor en la celda A1 un número y pulse (Intro) o desplácese directamente a la celda A2 en la que introduciremos un contenido alfanumérico tecleando 'HOLA.

En la parte superior derecha vemos la palabra ACTIVO, nos indica la modalidad en que se encuentra la hoja, en esta ocasión dispuesta a que introduzcamos valores en las celdas. Al introducir el con-

tenido de la celda A1 la palabra cambió a VALOR y en la celda A2 a ROTULO.

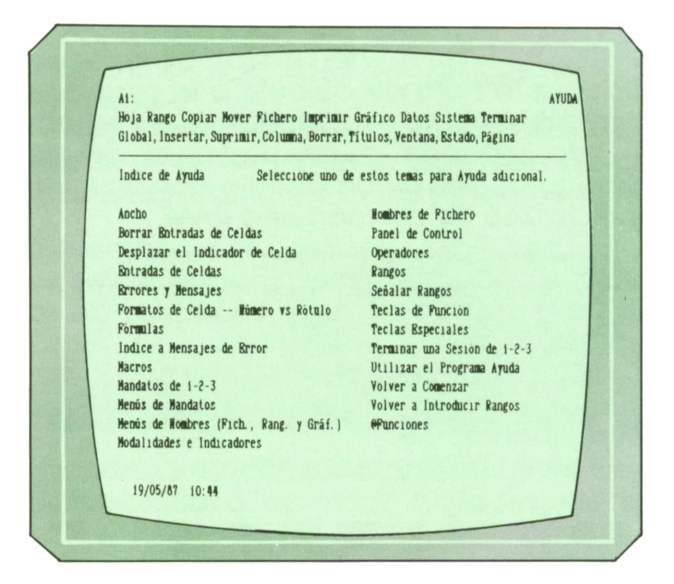
Abajo, a la izquierda, aparece la fecha y la hora del sistema, que va cambiando conforme lo hace ésta.

La sencillez de esta pantalla puede asustar en un principio sobre todo por la ausencia de menú. Sin embargo, éste existe en el Lotus 1-2-3; para que aparezca en la pantalla tecleamos (/) la barra inclinada a la derecha.

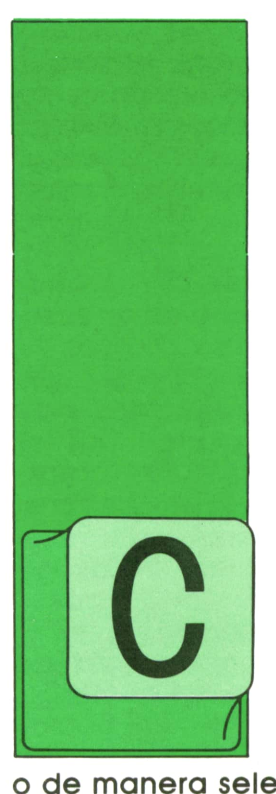


Observe que la palabra de la esquina superior derecha ha cambiado, ha dejado el modo activo para pasar al modo MENU. Para seleccionar una opción del menú puede desplazarse por el mismo con las teclas de cursor seleccionando el comando adecuado y pulsando (Intro) o simplemente pulsando la inicial del comando elegido. Al desplazarse con las teclas de cursor la línea inferior del menú va cambiando e indicando el contenido del comando sobre el que se encuentra el cursor.

Además del menú que puede resultar de gran ayuda, 1-2-3 dispone de un completísimo programa de ayuda al que se accede pulsando la tecla F1 y del que sale pulsando la tecla (Esc) o (Intro), volviendo al punto de trabajo en el que se encontraba. Consúltelo, comprobará cómo puede resultarle muy útil para su aprendizaje.



PASCAL



CON las estructuras de control que ya conocemos somos capaces de hacer que un conjunto de instrucciones se pueda ejecutar en secuencia, de manera repetitiva o de manera selectiva, según un criterio dado. Con esto es posible, en teoría, construir cualquier programa.

Sin embargo, hay ciertas estructuras construibles, por supuesto, en base a las estudiadas, que son tan frecuentes que merece la pena disponer de una forma específica de escribirlas, para así conseguir una mayor claridad en los programas.

Vamos a estudiar a continuación una de ellas:



La estructura case

Esta estructura podría considerarse como una estructura IF-THEN-ELSE especial. IF, como sabemos, permite escoger entre un máximo de dos posibilidades.

En multitud de ocasiones, sin embargo, es preciso escoger entre más de dos alternativas. Recordemos el procedimiento WriteDia:

```
procedure WriteDia (D: DiaDeLaSemana_t);
(* Presenta en pantalla el valor de D *)

 begin
  if D = Lunes then write ('Lunes')
   else if D = Martes then write ('Martes')
    else if D = Miercoles then write ('Miércoles')
     else if D = Jueves then write ('Jueves')
      else if D = Viernes then write ('Viernes')
       else if D = Sabado then write ('Sábado')
        else write ('Domingo')
 end;
```

(también se podría haber escrito agrupando los diferentes IF de esta otra manera:

```
if D = Lunes       then write ('Lunes');
if D = Martes      then write ('Martes');
if D = Miercoles  then write ('Miércoles');
...
```

que es algo menos eficiente, pues si, por ejemplo, D fuera Lunes, tras escribir «Lunes» se pasaría a comprobar si D es Martes, etc., cosa que no sucede con el procedimiento en su forma original).

Los casos como éste en que hay que escoger entre más de una alternativa según un criterio dado son muy frecuentes. Para esto está pensada la estructura CASE, que permite escoger entre varias instrucciones según sea el resultado de una expresión dada; el tipo resultante de esta expresión puede ser Integer, Char o cualquier otro tipo escalar creado por nosotros. WriteDia se escribiría así:

```
procedure WriteDia (D: DiaDeLaSemana_t);
(* Presenta en pantalla el valor de D *)

 begin
  case D of
    Lunes     : write ('Lunes');
    Martes    : write ('Martes');
    Miercoles: write ('Miércoles');
    Jueves    : write ('Jueves');
    Viernes   : write ('Viernes');
    Sabado    : write ('Sábado');
    Domingo   : write ('Domingo')
   end
 end;
```

La expresión (en este caso la variable D) cuyo valor controla la decisión a tomar se escribe entre las palabras reservadas CASE y OF; tras esto se escriben las diferentes instrucciones a escoger separadas entre sí por punto y coma.

Delante de cada instrucción y separada por dos puntos debe escribirse la lista de uno o más valores de la expresión (separados por comas) para los que deba ejecutarse, teniendo en cuenta que un valor dado sólo puede aparecer en a lo sumo una lista.

Para terminar, la estructura CASE se escribe la palabra reservada END, tras la que podrían venir otras instrucciones separadas por punto y coma.

Hay que considerar qué sucede cuando el resultado de la expresión no se encuentra en ninguna de las listas. En las primeras versiones de PASCAL se producía un mensaje de error y el programa se detenía; las versiones posteriores simplemente no ejecutan ninguna de las instrucciones de la estructura CASE y pasan a lo que venga a continuación.

La mayoría de los compiladores de PASCAL actuales permiten, sin embargo, definir opcionalmente una instrucción alternativa que se ejecutaría en estos casos; para ello se escribe esta instrucción antes de la palabra END final precedida de la palabra reservada ELSE (en algunos compiladores se emplea la palabra OTHERWISE; estamos ante algo no estándar del PASCAL y de ahí la posibilidad de que no en todos se utilice la misma palabra; en las más recientes proposiciones de estandarización del PASCAL se incluyen algunas ideas similares).

Igual que sucede con la estructura IF, las instrucciones a escoger pueden ser simples, como en el siguiente ejemplo, o de cualquier otro tipo (REPEAT, FOR...), incluso secuencias de instrucciones:

```
program AnalizaTexto;
(*----------------------------------------------------------*)
(* Analiza los caracteres de un texto introducido por teclado *)
(* y da el total de vocales, consonantes, cifras, etc.        *)
(*----------------------------------------------------------*)

const
 MaxLetras = 80;   (* Máximo número de letras por texto. *)

type
 Indice_t = 1..MaxLetras;

var
 Texto     : array [Indice_t] of char;
 Indice    : Indice_t;

 (* Variables usadas como contadores: *)

 Vocales,
 Consonantes,
 Cifras,
 OtroTipo : integer;
```

```
(*-----------------------------------------------*)
function  Mayuscula (C: char): char;
 begin
  if ('a' <= C) and (C <= 'z') then
      Mayuscula:= chr (ord(C)-ord('a')+ord('A'))
    else
      if C='ñ' then Mayuscula:= 'Ñ'
               else Mayuscula:= C
 end;
(*-----------------------------------------------*)

begin
 write ('Introduzca el texto.  ');
 writeln ('(Máximo ',MaxLetras,' caracteres)');
 readln (Texto);

 Vocales     := 0;
 Consonantes:= 0;
 Cifras      := 0;
 OtroTipo    := 0;

 for Indice:= 1 to MaxLetras do
   case Mayuscula (Texto [Indice]) of

     'A','E','I','O','U'  :  Vocales:= Vocales + 1;

     'B','C','D','F','G','H',
     'J','K','L','M','N','Ñ',
     'P','Q','R','S','T','V',
     'W','X','Y','Z'      :  Consonantes:= Consonantes + 1;

     '0','1','2','3','4',
     '5','6','7','8','9'  :  Cifras:= Cifras + 1;

     else  OtroTipo:= OtroTipo + 1
    end;

 writeln;
 writeln ('VOCALES      ',Vocales      :3);
 writeln ('CONSONANTES ',Consonantes :3);
 writeln ('CIFRAS       ',Cifras       :3);
 writeln ('DE OTRO TIPO',OtroTipo     :3)
end.
```

Para los ordenadores que permiten utilizar vocales acentuadas, éstas son unos caracteres totalmente distintos de las no acentuadas, por lo que pasarían a incrementar el contador OtroTipo si se encontraran en el texto. Pruebe el lector a modificar el programa para que contabilice por separado la letras acentuadas, los signos de puntuación, etc.

Ya hemos dicho que, normalmente, cuando se teclea un texto de menor longitud que el "array of char" al que va destinado, la variable se rellena por la derecha con los espacios en blanco (o caracteres cuyo ordinal es 0, según el compilador) que hagan falta; por ello, normalmente el contador OtroTipo tendrá un valor aparentemente exagerado; vamos a modificar el programa para que solucione este problema utilizando una función que, aplicada a un texto, devuelva su longitud real. Además, haremos que se puedan procesar varios textos consecutivos hasta que se introduzca uno en blanco, mostrándose tras cada uno sus cuentas y las cuentas totales considerando las de los anteriores.

```
program AnalizaVarios;
(*-----------------------------------------------------------*)
(* Analiza los caracteres de los textos introducidos por     *)
(* teclado y da las cuentas parciales y totales de vocales, *)
(* consonantes, cifras, etc.                                 *)
(*-----------------------------------------------------------*)
```

```
const
 MaxLetras = 80;   (* Máximo número de letras por texto. *)

type
 Indice_t  = 1..MaxLetras;
 Texto_t   = array [Indice_t] of char;

var
 Texto     : Texto_t;
 Indice    : Indice_t;

 (* Variables usadas como contadores: *)

 Vocales,       (* para la cuenta parcial    *)
 Suma_Voc,      (* para la cuenta acumulada *)
 Consonantes,   (* etcétera...               *)
 Suma_Con,
 Cifras,
 Suma_Cif,
 OtroTipo,
 Suma_Otr : integer;

(*-----------------------------------------------*)
function  Longitud (T: Texto_t): Indice_t;

(* Devuelve la longitud real del texto T *)

  var I: integer; Parar: boolean;
 begin
  I:= MaxLetras;
  Parar:= false;

  (* Se explora el texto de atrás para adelante *)
  (* hasta encontrar el primer carácter válido: *)

  repeat
    if I >= 1 then
        Parar:= (T[I] <> ' ') and (T[I] <> chr(0))
      else
        Parar:= true;
    if not Parar then I:= I-1
  until Parar;

  Longitud:= I
 end;

(*-----------------------------------------------*)
function  Mayuscula (C: char): char;
 begin
  if ('a' <= C) and (C <= 'z') then
      Mayuscula:= chr (ord(C)-ord('a')+ord('A'))
    else
      if C='ñ' then Mayuscula:= 'Ñ'
               else Mayuscula:= C
 end;
(*-----------------------------------------------*)

begin
 Suma_Voc:= 0;
 Suma_Con:= 0;
 Suma_Cif:= 0;
 Suma_Otr:= 0;
 repeat (* repetir el proceso de textos *)

  writeln;
  writeln ('Texto:');
  readln (Texto);

   (* Poner a cero los parciales: *)

   Vocales     := 0;
   Consonantes:= 0;
   Cifras      := 0;
   OtroTipo    := 0;

  for Indice:= 1 to Longitud (Texto) do
    case Mayuscula (Texto [Indice]) of

      'A','E','I','O','U'  :  Vocales:= Vocales + 1;

      'B','C','D','F','G','H',
      'J','K','L','M','N','Ñ',
      'P','Q','R','S','T','V',
      'W','X','Y','Z'       :  Consonantes:= Consonantes + 1;
```

```
     '0','1','2','3','4',
     '5','6','7','8','9' :  Cifras:= Cifras + 1;

     else  OtroTipo:= OtroTipo + 1
    end;

(* Enseñar los parciales: *)

write   ('     Vocales:     ',Vocales      :3);
write   ('     Consonantes:',Consonantes :3);
write   ('     Cifras:      ',Cifras       :3);
writeln ('     Otras:       ',OtroTipo     :3);

(* Acumularlos: *)

Suma_Voc:= Suma_Voc + Vocales;
Suma_Con:= Suma_Con + Consonantes;
Suma_Cif:= Suma_Cif + Cifras;
Suma_Otr:= Suma_Otr + OtroTipo;

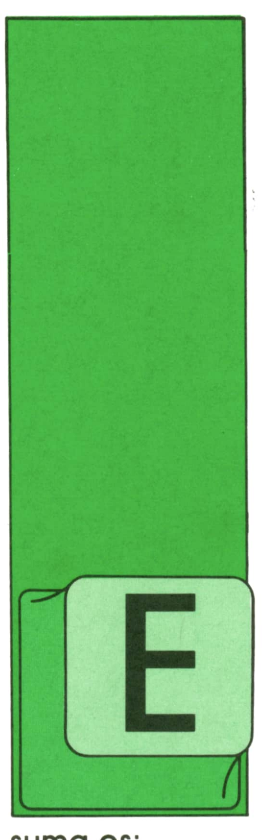
(* Enseñar los totales: *)

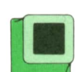
  write   ('Totales --->     ',Suma_Voc :3);
  write   ('                 ',Suma_Con :3);
  write   ('                 ',Suma_Cif :3);
  writeln ('                 ',Suma_Otr :3)

 until Longitud (Texto) = 0
end.
```

OTROS LENGUAJES

COBOL





Instrucciones aritméticas

En todo programa se realizan cálculos con los datos. Seguidamente se expone el formato de las sentencias de operación más comunes.

La instucción de suma es:

ADD { literal numérico / campo-numerico-1 } TO campo-numérico-2

Los literales y campos que aparecen a la izquierda de la partícula TO se suman al campo que está a su derecha, dejando el resultado en éste.

Si en el programa existe esta secuencia de instrucciones:

```
MOVE 3 TO A.
MOVE 10 TO B.
ADD 2 A TO B.
```

El valor final de B es 15 (2 + 3 + 10). El contenido de A permanece invariable.

Para restar se emplea la instrucción:

SUBTRACT { literal-numérico / campo-numérico-1 } FROM campo-numérico-2

Se suman los literales y valores de los campos que se encuentren a la izquierda del FROM, restándose a su vez el contenido de la variable de la derecha.

Después de la serie de instrucciones:

```
MOVE 3 TO A.
MOVE 10 TO B.
SUBTRACT 2 A FROM B.
```

B toma el valor 5 (10 − (2 + 3)).

La multiplicación tiene el siguiente formato:

MULTIPLY { campo-numérico-1 / literal } BY campo-numérico-2

Se multiplican los dos valores y el resultado se almacena en el último campo (campo-numérico-2).

```
MOVE 3 TO A.
MOVE 10 TO B.
MULTIPLY A BY B.
```

El valor final de B será 30 (3*10).

Para dividir se utilizan las instrucciones:

DIVIDE { campo-numérico-1 / literal } INTO campo-numerico-2

Se divide el valor del campo-2 por el valor de campo-1 o el literal, almacenando en campo-2 el resultado.

```
MOVE 12 TO B.
DIVIDE 3 INTO B.
```

Después de ejecutar las instrucciones B, tendrá como valor 4.

Pero en un programa pueden surgir expresiones aritméticas mucho más complicadas, resultando muy pesada su representación con las instrucciones anteriores. Para estas ocasiones existe una nueva instrucción.

COMPUTE campo-1 = expresión aritmética.

Los operadores que se utilizan en una expresión aritmética son:

+ Sumar.
- Restar.
* Multiplicar.
/ Dividir.
** Potenciación.
() Paréntesis.

Existe una jerarquía de operadores, a la hora de trabajar con ellos.

— Se ejecutan primero las operaciones encerradas entre paréntesis.

— Seguidamente se calcula el signo que afecta a la expresión.
— El operador de potenciación.
— Multiplicación y división.
— Suma y resta.

Dentro de la misma prioridad se ejecutan antes los que estén más a la izquierda. Hay que colocar los paréntesis de forma estratégica para romper el orden de prioridades, según interese.

El siguiente programa pide por pantalla el código de un empleado, su sueldo, el porcentaje de gratificación y la cantidad a descontar. Se mostrará por pantalla el total que debe percibir cada empleado.

La fórmula que se aplica para obtener el sueldo neto es:

$$TOTAL = SUELDO + SUELDO * \frac{GRATIF}{100} - DESCUENTO$$

```
IDENTIFICATION DIVISION.
 PROGRAM-ID. EJ-OPERACION.

ENVIRONMENT DIVISION.

DATA DIVISION.


 WORKING-STORAGE SECTION.

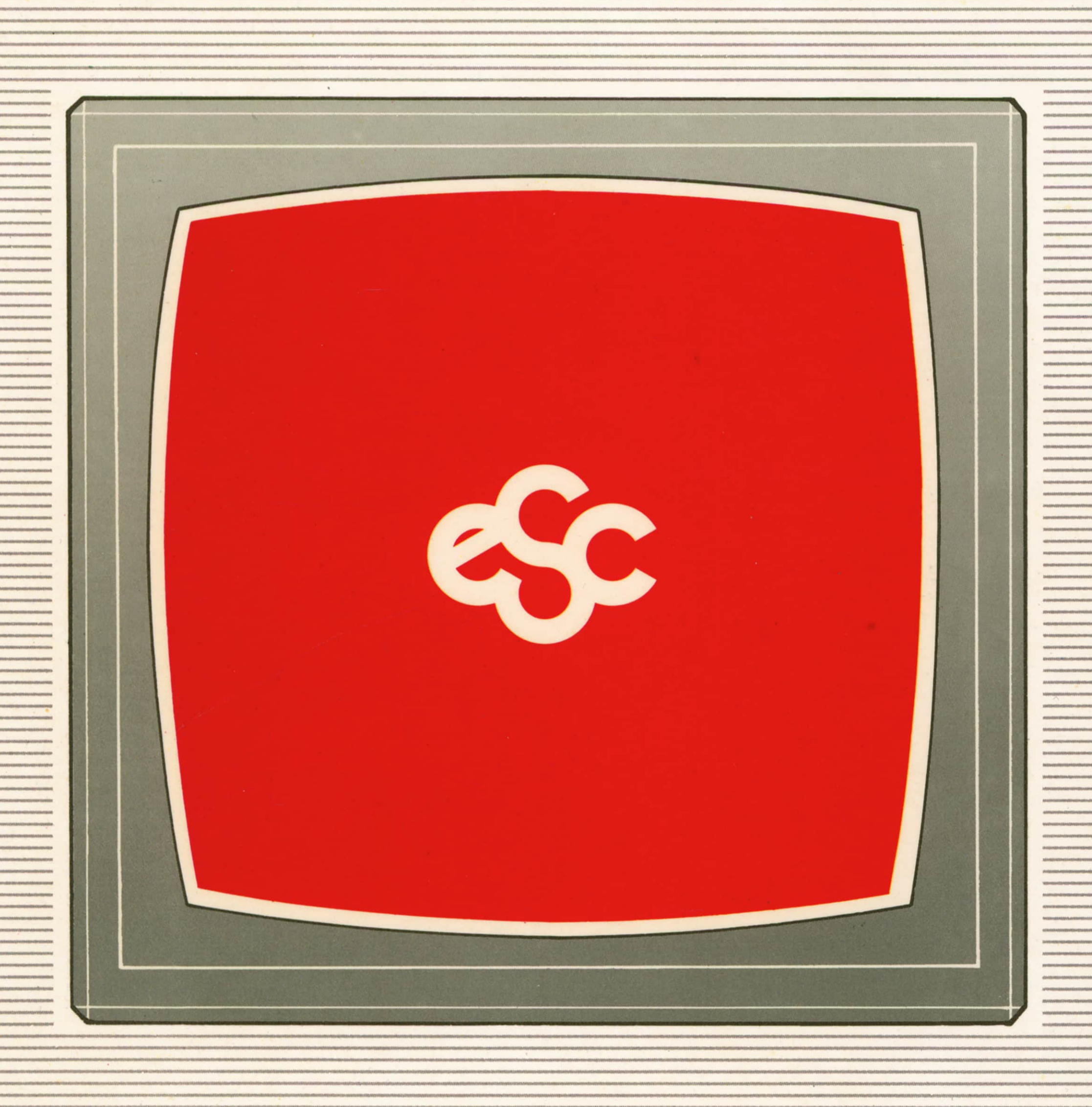
  01 REG-EMPLEADO.
     05 CODIGO                    PIC X(5).
     05 SUELDO                    PIC 9(6).
     05 GRATIF                    PIC 99.
     05 DESCUENTO                 PIC 9(5).

  01 TOTAL                        PIC 9(7).


PROCEDURE DIVISION.

INICIO.
       DISPLAY 'DATOS DE EMPLEADO'.
       DISPLAY 'CODIGO'.
       ACCEPT CODIGO.
       DISPLAY 'SUELDO'.
       ACCEPT SUELDO.
       DISPLAY 'GRATIFICACION'.
       ACCEPT GRATIF.
       DISPLAY 'DESCUENTO'.
       ACCEPT DESCUENTO.
       COMPUTE TOTAL = SUELDO + SUELDO * GRATIF / 100
                      - DESCUENTO.
       DISPLAY CODIGO ' COBRARA ' TOTAL.

FIN-PROGRAMA.
       STOP RUN.
```



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼